Contents

Part I: The Python Language

Lesson 1. Vocabulary

A **module** is file that contains python code. On your hard drive, this is saved as myfile.py.

Lesson 2. Loading Modules

2.1 Module Path Needs to be Specified

ref

```
import tkinter as tk
import sys

# append the full path of the folder
sys.path.append('/Users/donaldelger/SpyderProjects/Reflection/code')

import reflect

if __name__ == '__main__':
    root = tk.Tk()
    r1 = reflect.Reflect(root)
    r1.grid(padx=15, pady=15)
    root.geometry('+20+20')
    root.mainloop()

\chapter{Round}

\subsection{Round to a Specified Number of Digits}


\begin{lcverbatim}
In []: round(129.4375)
Out[]: 129

In []: round(129.4375, 2)
Out[]: 129.44
\end{lcverbatim}


\cverb!round(number[, ndigits])! Round to ndigits (default=0) digits after the decimal point. The 0.5



\subsection{Round Up to Nearest Whole Number}

The code
\begin{codeA}{}{Python}
import math
val = math.ceil(2.4)
```

will set `val = 2`. Note that the type of val is float, not integer. The method `math.ceil(x)` returns the smallest integer value that is greater than or equal to x.

Lesson 3. Simple Math

3.1 Mean

```
import numpy as np
gap = [3.27, 3.148, 3.230, 3.182, 3.257, 3.195, 3.262, 3.244, 3.161, 3.222,
       3.24, 3.207]
print(np.mean(gap))
```

```
list1 = [2.4, 7.6, 9.3]
mean1 = sum(list1) / float(len(list1))
```

Lesson 4. The String

4.1 Slice

This **Digital Oceans** article explain how to slice a string.

To include the front or back end of a string, omit one of the numbers is the `string[n:n]` syntax.

Example. `print(mystr[:10])` prints the first 10 characters.

Example: `print(mystr[10:])` prints from character 10 to the end.

4.2 Capitalize or Lower Case

```
ms = 'hello'

ms.title()  # returns Hello
ms.capitalize()  # returns Hello
ms.upper ()  # returns HELLO
ms[:1].upper() + ms[1:]  # returns Hello
```

4.3 Split a string

Example: Splitting a file name into parts

```
>>> t = '10t.jpg'
>>> t.split('.')
['10t', 'jpg']
```

Example: Splitting a sentence into words

```
>>> t = 'the big dog ran fast'
>>> t.split()
['the', 'big', 'dog', 'ran', 'fast']
```

4.4 String to List: Split at LineBreaks

tutorialspoint

Ref.

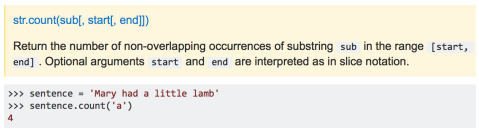
```
str.splitlines(True)  # Include \n
str.splitlines()      # Exclude \n
```

4.5 Search and Replace

Example. If `a = '1:2,3'`, then `a.replace(':', ", ")` gives the string `'1, 2, 3'`.

The syntax is `mystr.replace(old, new[, max])`. If max is omitted, all instances of the string are replaced. If max is present, then up to max instances are replaced.

4.6 Count Occurrences



4.7 String to Numbers

`eval(sting)`

```
>>> a = '(1, 2, 3)'
>>> eval(a)
(1, 2, 3)
```

4.8 Execute a Statement written as a String

`exec(sting)`

4.9 Remove Leading and Trailing Characters: Esp. Whitespace

`mystring.strip()` : remove leading and trailing whitespace
`mystring.strip('2')` : remove leading and trailing 2s
`mystring.lstrip()` : remove leading whitespace
`mystring.rstrip()` : remove trailing whitespace

The website **tutorialspoint** describes the strip() method.

The website **datascience** describes the strip(), lstrip(), and the rstrip() methods.

4.10 Check if String is Empty

If you know that your variable is a string, then use
`if not string_var:` .

If your variable could also be some other type, then use
`if myString == "":`

Lesson 5. The String Format Method

5.1 Format Specifier

:xx.xx see table for all the possible x values

The general form of a *standard format specifier* is:

```
format_spec     ::=  [[fill]align][sign][#][0][width][grouping_option][.precision][type]
fill            ::=  <any character>
align           ::=  "<" | ">" | "=" | "^"
sign            ::=  "+" | "-" | " "
width           ::=  digit+
grouping_option ::=  "_" | ","
precision       ::=  digit+
type            ::=  "b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s" | "x" | "X" | "%"
```

Figure 5.1

5.2 Older ...

resource
euro resource

The SFM is way to insert data into a string of characters so that the resulting string effectively communicates.

5.3 Numbers and Currency

`'{:.2f}'.format(3.141592653589793)` gives 3.14

To print $1,234.5, use the command `'${:,.2f}'.format(1234.5)`

```
1 power = 170935020100
2 print('power is {:.4e}'.format(power))
3
4 # output: power is 1.7094e+11
```

5.4 Values and Key=Value Information

The general format is
`mystring.format(po, p1, p2, ... , ko = vo, k1 = v1, ....)`
where p=parameter and k=v means that keyword=value

5.5 Use a Dictionary

Example:

```
geopoint = {'latitude':41.123,'longitude':71.091}
print('{latitude} {longitude}'.format(**geopoint))
```

The structure is:

```
ddd = {key1:value2, key2: value2, ...}
string.format(**ddd)
```

Lesson 6. The List

6.1 Remove or Pop item

```
In: a = [1, 'dog', 2.4]
In:  a.remove(1)
In: a
Out: ['dog', 2.4]
In: a.remove('dog')
In: a
Out: [2.4]
```

The list method `pop` removes the item at index i from the list while also returning the item:

```
In[1]: mlist = [0, 1, 2, 3]

In[2]: mlist.pop(2)
Out[2]: 2

In[3]: mlist
Out[3]: [0, 1, 3]
```

6.2 Add Item at Start or Before an Index

Use the list insert method.

Example. `a.insert(0, 0.42)` adds an item at the start of the listl

Example: For the list `a = [0, 2.4, -9.3]`, the command `a.insert(1, 0.42)` gives `[0, 0.42, 2.4, -9.3]`

6.3 Convert Between Tuples and Lists

```
a = (1, 2, 3)
b = list(a)
```

```
b = [1, 2, 3]
c = tuple(b)
```

6.4 The List Comprehension

The list comprehension is a method to construct lists without using loops; see **this article**; see **python for beginers**

```
new_list = [expression(i) for i in old_list if filter(i)]
```

Example

```
squares = [x**2 for x in range(10)]
```

6.5 List: integers to float

```
>>> a = (1, 2, 3)
>>> [float(i) for i in a]
[1.0, 2.0, 3.0]
```

Lesson 7. The Dictionary

7.1 Definition

A dictionary is a type of data structure that is made up of a collection or set of key and value pairs.

For example, suppose you were calculating your monthly expenses, you might have some data like this

Table 7.1: Expense data in units of dollars/month

Key (name of category)	Value (amount of category)
utilities	106
rent	632
groceries	282

The dictionary is the type of DS that is best for storing this type of data; shows up all the time; this is why the dictionary is super useful

7.2 Creating a Dictionary

One way is
`dictionary_name = {key:value, key:value, ...}`

7.3 Remove an Item

```
cell_dict.pop('pk')
```

7.4 Getting a list of the keys

`book.dct.keys()` Returns a list containing the keys

7.5 Dictionary

The string
`"{'seq': 2, 'desc': 'good', 'a': 'dog'}"`

was created using

```
dict = {"a":"dog", "desc":"good", "seq":2}
a = str(dict)
```

To covert the string `a` back to a dictionary use the `eval()` command as follows: `print(eval(a)`

7.6 Dictionary from Two Lists

List 1: `a = [9, 'dog', 'h']`
List 2: `b = ['wookie', 22.4 , 'general']`

Command: `dict(zip(a,b))`
note that zip is a built in function

Result:
`{9: 'wookie', 'dog': 22.4, 'h': 'general'}`

7.7 Reverse Lookup

given the value find the key: dictionary not intended for this

one approach is to build a reverse dictionary

```
1 reverse_dict = {}
2 for (key, value) in my_dict.items():
3   reverse_dict[value] = key
```

Lesson 8. Copy and Deep Copy

from copy import copy, deepcopy

When you assign dict2 = dict1, you are not making a copy of dict1, it results in dict2 being just another name for dict1.

To copy the mutable types like dictionaries, use copy / deepcopy of the copy module.

ref.

Lesson 9. Loops

break: end a loop

continue: ....

syntax

```
if expression1:
   statement(s)
elif expression2:
   statement(s)
elif expression3:
   statement(s)
else:
   statement(s)
```

9.1 Using a Dictionary

For dictionary `d` :
`for key, value in d.items()`

9.2 List and Counter: Enumerate

The pythonic way is to use `enumerate` :

```
for idx,item in enumerate(list):
```

9.3 List Comprehension

Lesson 10. Logic, Conditionals, etc.

```
if condition:
    [block of code]
```

```
if condition:
    [block of code]
else:
  [block of code]
```

The six comparison operators:

```
== !=
< <=
> >=
```

¡ ¡=

True or False

Unfortunately it is not as easy in real life as it is in Python to differentiate between true and false:
The following objects are evaluated by Python as False:

- numerical zero values (0, 0L, 0.0, 0.0+0.0j),
- the Boolean value False,
- empty strings,
- empty lists and empty tuples,
- empty dictionaries.
- plus the special value None.

Abbreviated IF statement

C programmers usually know the following abbreviated notation fo
the if construct:

```
max = (a > b) ? a : b;
```

This is an abbreviation for the following C code:

```
if (a > b)
   max=a;
else
   max=b;
```

C programmers have to get used to a different notation in Python:

```
max = a if (a > b) else b;
```

Lesson 11. Function

A function is a collection of program instructions that are packaged as a unit and that do one job. A function is also called a method, a subroutine, a routine, and a subprogram.

An argument is an input that is passed to a function.

The docstring is ...

A parameter is a piece of information that the function needs to do its job.

A positional argument is ..

A keywork argument is ...

11.1 *arg and **kwargs

*arg and **kwargs

```
def dog(**kwargs):
    print(kwargs, type(kwargs))

kwargs = {'one': 22, 'two': 33}
dog(**kwargs)

def rat(*args):
    print(args, type(args))

mylist = [7, 'hat', 9.43]
rat(*mylist)
```

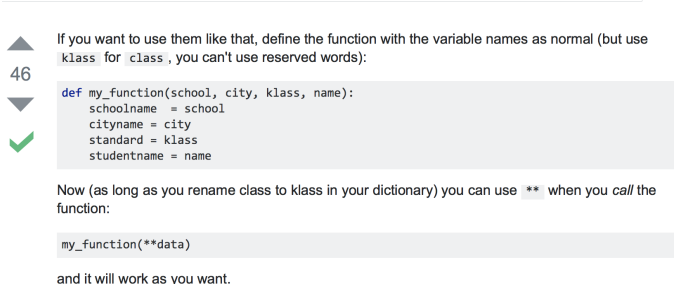
another example

```
def dog(**kwargs):
    for k,v in kwargs.items():
        print(k, v)

mydict = {'a':10, 'b':'rat'}
dog(a = 10, b = 'rat')
dog(**mydict)
```

This will print

```
b rat
a 10
b rat
a 10
```

11.2 Passing a Dictionary to a Function



46

If you want to use them like that, define the function with the variable names as normal (but use `klass` for `class`, you can't use reserved words):

```
def my_function(school, city, klass, name):
    schoolname  = school
    cityname = city
    standard = klass
    studentname = name
```

Now (as long as you rename class to klass in your dictionary) you can use `**` when you *call* the function:

```
my_function(**data)
```

and it will work as you want.

Passing a dictionary and setting default values if not passed within the dictionary; very clever. Note that the second argument to `dict.pop()` is the default value if the key is not found in the dictionary.

```
1   def __init__(self, master=None, **kw):
2       """
3       WIDGET-SPECIFIC OPTIONS
4
5           locale, firstweekday, year, month, selectbackground,
6           selectforeground
7       """
8       # remove custom options from kw before initializating ttk.Frame
9       fwday = kw.pop('firstweekday', calendar.MONDAY)
10      year = kw.pop('year', self.datetime.now().year)
11      month = kw.pop('month', self.datetime.now().month)
12      locale = kw.pop('locale', None)
13      sel_bg = kw.pop('selectbackground', '#ecffc4')
14      sel_fg = kw.pop('selectforeground', '#05640e')
```

Lesson 12. Lambda (Anonymous) Function

This blog post explains the lambda function.

`lambda b, c : b*c` has arguments (b,c) and returns a value of b*c.

A lambda function does not need parameters. For example, this code

```
b = 9
k = lambda: b+4.2
print(k())
```

returns 13.2.

A lambda function without parameters is used by this lambda function to pass arguments in the context of tkinter:

```
lt = Label(c3,text='Controller')
lt.pack(side='top',expand=1,fill='x')
l = Label(c3,text='',bg='red')
l.pack(side='top',expand=1,fill='x')
lt2 = Label(c3,text='Velocity')
lt2.pack(side='top',expand=1,fill='x')
l2 = Label(c3,text='',bg='yellow')
l2.pack(side='top',expand=1,fill='x')
lt3 = Label(c3,text='Desired Heading')
lt3.pack(side='top',expand=1,fill='x')
l3 = Label(c3,text='',bg='blue')
l3.pack(side='top',expand=1,fill='x')

b2 = Button(c3,text='load',command=lambda:loadme(l,l2,l3))
b2.pack(fill='x', padx=2, pady=2)
root.mainloop()
```

Lesson 13. File Read and Write

Ref.

13.1 Read a File

```
with open(file_name) as file_object:
    str = file_object.read()
```

```
fn = ("test.txt", "w")
fn.write("I am a test file.\n")
fn.write("Line2 ...")
fn.close()
```

13.2 Read a File by Line

This **source** states that the best way to read a file line by line is as follows

```
with open(...) as f:
    for line in f:
        <do something with line>
```

The command `file_object.readlines()` will read a text file by line.

In the following code, I strip off the white space using the `rstrip()` method as shown.

```
fn = ('my_file.txt')
with open(fn) as file_object:
    task_list = [line.rstrip() for line in file_object.readlines()]
```

13.3 Modes

- '*r*' – Read mode which is used when the file is only being read
- '*w*' – Write mode which is used to edit and write new information to the file (any existing files with the same name will be erased when this mode is activated)
- '*a*' – Appending mode, which is used to add new data to the end of the file; that is new information is automatically amended to the end
- '*r+*' – Special read and write mode, which is used to handle both actions when working with a file

Character	Meaning
`'r'`	open for reading (default)
`'w'`	open for writing, truncating the file first
`'x'`	open for exclusive creation, failing if the file already exists
`'a'`	open for writing, appending to the end of the file if it exists
`'b'`	binary mode
`'t'`	text mode (default)
`'+'`	open a disk file for updating (reading and writing)
`'U'`	universal newlines mode (deprecated)

The default mode is `'r'` (open for reading text, synonym of `'rt'`). For bin

13.4 Stripping Whitespace

1135

Try the method `rstrip()` (see doc Python 2 and Python 3)

```
>>> 'test string\n'.rstrip()
'test string'
```

Python's `rstrip()` method strips *all* kinds of trailing whitespace by default, not just one newline as Perl does with `chomp`.

```
>>> 'test string \n \r\n\n\r \n\n'.rstrip()
'test string'
```

To strip only newlines:

```
>>> 'test string \n \r\n\n\r \n\n'.rstrip('\n')
'test string \n \r\n\n\r '
```

There are also the methods `lstrip()` and `strip()`:

```
>>> s = "   \n\r\n  \n  abc   def \n\r\n  \n  "
>>> s.strip()
'abc   def'
>>> s.lstrip()
'abc   def \n\r\n  \n  '
>>> s.rstrip()
'   \n\r\n  \n  abc   def'
```

13.5 Writing a File

```
with open(fname, "w") as fobj:
        fobj.write("hello world"
```

Lesson 14. Read Comma Separated Variable (csv)

Lesson 15. Dump and Load Data to a File

To **serialize** data is to convert a data structure to a format that is consistent with how data is stored on the hard drive.

15.1 JSON

best approach: use dump and load

file: json1.py

```
import json

pets = {'cat': 'Tabby', 'dog':'Hoshi'}

# ====== dump string (dumps and loads) ==========
with open('pet_list.txt', 'w') as file:
      file.write(json.dumps(pets))

with open('pet_list.txt', 'r') as file:
      my_pets = json.loads(file.read())

# === dump file like object (dump and load) =====
with open('pet_list2.txt', 'w') as fp:
    json.dump(pets, fp)

with open('pet_list2.txt', 'r') as fp:
    d9 = json.load(fp)
```

15.2 Pickle (Binary)

223

What are you going to do with the file? Does this file exist for humans, or other programs with clear interoperability requirements, or are you just trying to serialize a list to disk for later use by the same python app. If the second case is it, you should be pickleing the list.

```
import pickle

with open('outfile', 'wb') as fp:
    pickle.dump(itemlist, fp)
```

To read it back:

```
with open ('outfile', 'rb') as fp:
    itemlist = pickle.load(fp)
```

Lesson 16. Work with Times and Dates

Guru99

time vs datetime vs timestamp ...

16.1 Getting Started

Import the module (recommendation):
`from datetime import datetime, timedelta`

`datetime.today()` Create a DO for now

`datetime(2016, 12, 25)` Create a DO for a specified date

`date1 + timedelta(days=1)` : Add one day to a DO

16.2 Delta Time

datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)

a. Speed: Minutes/Mile

```
1 import datetime
2 d = datetime.timedelta(minutes=1, seconds=40.)
3 rate = d/(2/9.)
4 print(rate)
5 # A 0:07:30 pace
```

b. Time Between Two Dates

```
from datetime import date

d0 = date(2008, 8, 18)
d1 = date(2008, 9, 26)
delta = d0 - d1
print delta.days
```

16.3 String to DateTime

The command

```
datetime.datetime.strptime('8/22/16', '%m/%d/%y')
```

returns a datetime object for Aug. 22, 2016. The letters strp stand for *string parse*.

16.4 DateTime to String

The commands

```
1 today = datetime.datetime(2016, 12, 6)
2 today.strftime('Today is the %jth day of %Y')
```

give the string Today is the '341th day of 2016'.

Line 1 creates a datetime object for 12/6/2016. Then, line 2 creates a string that uses data from the string time object. By using the string directives any format can be created.

16.5 Directives for Strings

The table that follows lists some of the directives. More directives are described by **Tutorials Point**

Directive	Meaning
%Y	year as in 2016
%y	year as in 16
%m	month as a 2 digit number
%B	month name as in December
%b	month name as in Dec
%d	day of the month (00 to 31)
%e	day of the month (1 to 31)
%j	day of the year (0 to 365)
%a	weekday name, 3 characters (Tue, Thu)
%A	weekday name, spelled out (Tuesday)

16.6 Get the ISO 8601 Date as a String

```
1 datetime.datetime.now().isoformat()
2 Out[53]: '2018-04-18T05:50:07.242489'
3
4 datetime.datetime.today().isoformat()
5 Out[54]: '2018-04-18T05:52:06.088015'
6
7 ds = datetime.datetime.now().isoformat()
8 ds = ds.split('T')[0]
9 Out[55]: '2018-04-18'
```

Lesson 17. File Operations

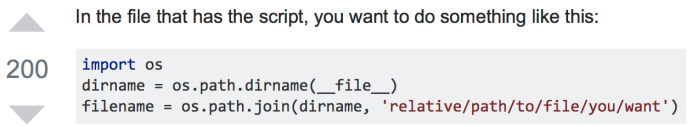
17.1 Get Folder Path

The command `os.path.dirname(path)` returns the pathname of the folder. The argument `path` can be a pathname of a file or folder.

17.2 Use Relative File Path

2018-10-05: tricky when running from terminal. The code that follows is what I did to get the folder name of the project that holds the python file in a directory py. Of course, this algorithm may need to be modified ...

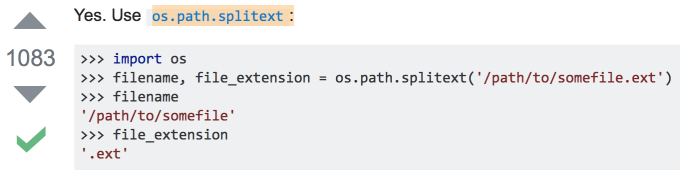
```
path = os.path.abspath(__file__)  # path to python script
dirname = os.path.dirname(os.path.dirname(path)) # root folder
```



17.3 Get Path + File

The command `path, file = os.path.split('/Users/donaldelger/Desktop/ 9.pdf')` returns /Users/donaldelger/Desktop and 9.pdf.

17.4 Split Path into Extension plus Other



17.5 Get File Name w/o Extension

pn = pathname of a file. The following returns the file name

`os.path.splitext(os.path.basename(pn))[0]`

```
1
2 pn = '/Users/donaldelger/Desktop/stove1.png'
3
4 os.path.basename(pn)
5 Out[16]: 'stove1.png'
6
7 os.path.splitext(fn)
8 Out[18]: ('stove1', '.png')
9
10 os.path.splitext(fn)[0]
11 Out[19]: 'stove1'
12
13 os.path.splitext(os.path.basename(pn))[0]
14 Out[20]: 'stove1'
```

17.6 Open Finder to Folder

`subprocess.run(["open", folder_path])`

17.7 Change Directory

The command `os.chdir(path)` changes the current working directory to the folder path in pathname path. Note that *path* can be pathname of a file or a folder.

17.8 Get the Current Working Directory

The command `os.getcwd()` returns the pathname of the current working directory.

17.9 Check File Modification Date

The command `os.path.getmtime(path)` gives the Unix timestamp of when the file at `path` was last modified.

17.10 Open a File

If something can be done with terminal, it can be done with the module subprocess.

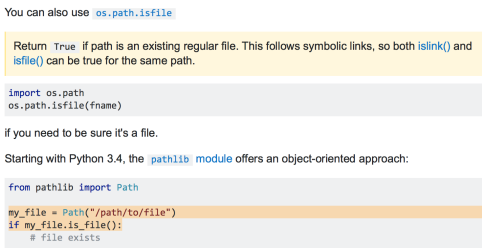
In terminal, the command `open ski.jpg` will open the image ski.jpg. In python, the following commands will open this file

```
1 import subprocess
2 # snip ....
3
4 subprocess.run(['open','ski.jpg'])
```

17.11 Check for File or Folder

For a directory, use `os.path.isdir(folder)`.

For a file or directory, use `os.path.exists(path)`


You can also use `os.path.isfile`

Return `True` if path is an existing regular file. This follows symbolic links, so both islink() and isfile() can be true for the same path.

```
import os.path
os.path.isfile(fname)
```

if you need to be sure it's a file.

Starting with Python 3.4, the `pathlib` module offers an object-oriented approach:

```
from pathlib import Path

my_file = Path("/path/to/file")
if my_file.is_file():
    # file exists
```

17.12 Write New File + Create All Folders in Path

```
filename = '/foo/bar/baz.txt'
os.makedirs(os.path.dirname(filename), exist_ok=True)
with open(filename, "w") as f:
    f.write("FOOBAR")
```

stack overflow

17.13 Make a Folder

For one folder use `os.mkdir(path)` For a folder plus all folders in the path, use `os.makedirs(path)`

17.14 Copy a File, Folder, Directory Tree

File: `shutil.copyfile(src, des)`
Folder or directory tree: `shutil.copytree(src, des)`

17.15 Open a File or Folder

Use `subprocess.run(["open", fname])`

17.16 Rename a File or Folder

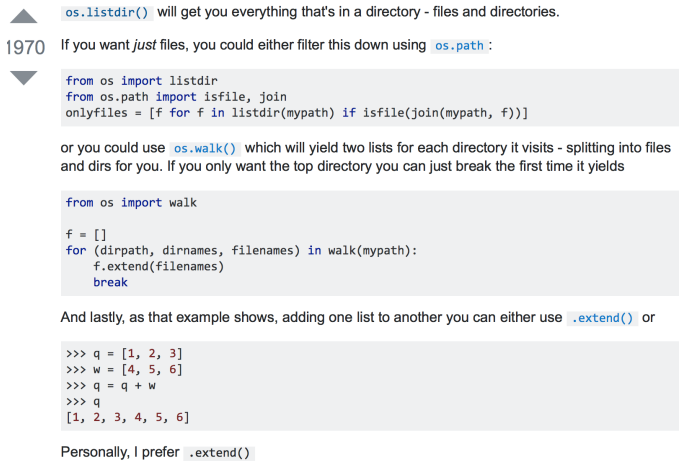
`os.rename(src, dest)`

17.17 Remove a Directory Tree

The command `shutil.rmtree(path)` removes a directory, the subdirectories, and the files. The argument `path` must be a directory.

17.18 List Files in a Directory

ref



1970

`os.listdir()` will get you everything that's in a directory - files and directories.

If you want *just* files, you could either filter this down using `os.path`:

```
from os import listdir
from os.path import isfile, join
onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f))]
```

or you could use `os.walk()` which will yield two lists for each directory it visits - splitting into files and dirs for you. If you only want the top directory you can just break the first time it yields

```
from os import walk

f = []
for (dirpath, dirnames, filenames) in walk(mypath):
    f.extend(filenames)
    break
```

And lastly, as that example shows, adding one list to another you can either use `.extend()` or

```
>>> q = [1, 2, 3]
>>> w = [4, 5, 6]
>>> q = q + w
>>> q
[1, 2, 3, 4, 5, 6]
```

Personally, I prefer `.extend()`

17.19 List Files in Dir + SubDirs

```
import os
my_dir = ('/Users/donaldelger  ... /tplates/task')
for root, directories, filenames in os.walk(my_dir):
#    for directory in directories:
#        print(os.path.join(root, directory))
    for filename in filenames:
        print(os.path.join(root,filename))
```

can also use glob

17.20 Building Absolute Path

killer !

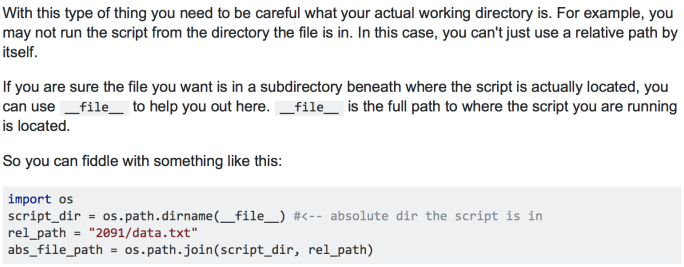
```
os.path.join(a, b)
```

17.21 Absolute File Path

Check for absolute path. Returns `True` if the path is an absolute path; `False` otherwise
`os.pth.isabs(my_path)`

Convert a relative path to an absolute path (works iff python working directory equals the current working directory. If the relative path in the current working directory is `'figs/9sit.tex'` , then the command to get the absolute path is
`os.path.abspath('figs/9sit.tex')`

17.22 Relative File Path



With this type of thing you need to be careful what your actual working directory is. For example, you may not run the script from the directory the file is in. In this case, you can't just use a relative path by itself.

If you are sure the file you want is in a subdirectory beneath where the script is actually located, you can use `__file__` to help you out here. `__file__` is the full path to where the script you are running is located.

So you can fiddle with something like this:

```
import os
script_dir = os.path.dirname(__file__) #<-- absolute dir the script is in
rel_path = "2091/data.txt"
abs_file_path = os.path.join(script_dir, rel_path)
```

17.23 Remove a File, Folder, or Directory

`os.remove()` will remove a file

`os.rmdir()` will remove an empty directory.

`shutil.rmtree()` will delete a directory and all its contents.

Lesson 18. Comparison Operators

Operation	Meaning	Notes
`<`	strictly less than	
`<=`	less than or equal	
`>`	strictly greater than	
`>=`	greater than or equal	
`==`	equal	
`!=`	not equal	(1)
`<>`	not equal	(1)
`is`	object identity	
`is not`	negated object identity	

For "not equal," the preferred operator is `=!`, `<>`.

18.1 Sorting

See **ref. 1**, and **ref. 2**

The built in function `sorted` returns a sorted iterable:

```
>>> sorted([5, 2, 3, 1, 4])
[1, 2, 3, 4, 5]
```

For a list, the method `sort` returns a list in place:

```
>>> a = [5, 2, 3, 1, 4]
>>> a.sort()
>>> a
[1, 2, 3, 4, 5]
```

Lesson 19. Exception Handling: Try and Except

Operational error did not work ...

```
try:
    print("Hello World")
except OperationalError:
    print("This is an error message!")
```

`OperationalError` : Errors which are related to MySQL's operations.

Lesson 20. Logging

Useful Resources about Logging

1. **Fang**
2. **Tutorial 1**

Fang Tutorial 1 and **Tutorial 2** and **logging cookbook**.

Level	When it's used
DEBUG	Detailed information, typically of interest only when diagnosing problems.
INFO	Confirmation that things are working as expected.
WARNING	An indication that something unexpected happened, or indicative of some problem in the near future (e.g. 'disk space low'). The software is still working as expected.
ERROR	Due to a more serious problem, the software has not been able to perform some function.
CRITICAL	A serious error, indicating that the program itself may be unable to continue running.

20.1 Concepts

To reset the logger, restart the kernal, or start a new iPython console, or restart Spyder.

20.2 Minimum Worked Example

```
import logging
logging.getLogger().setLevel(logging.INFO)
logging.debug('This message should appear on the console')
logging.info('So should this')
logging.warning('And this, too')
logging.error('error')
```

20.3 Setting the Logging Configuration

```
import logging
logging.basicConfig(filename='example.log',level=logging.DEBUG)
logging.debug('This message should go to the log file')
logging.info('So should this')
logging.warning('And this, too')
```

20.4 Customizing the Logging Message

```
logging.basicConfig(format='%(levelname)s: (%(asctime)s) '
                    '%(message)s', datefmt='%m/%d %I:%M')
logging.getLogger().setLevel(logging.DEBUG)
logging.debug('This message should appear on the console')
logging.info('So should this')
logging.warning('And this, too')
logging.error('error')
```

The output from the preceeding code is

```
DEBUG: (05/20 05:34) This message should appear on the console
INFO: (05/20 05:34) So should this
WARNING: (05/20 05:34) And this, too
ERROR: (05/20 05:34) error
```

20.5 Capturing Exceptions

add the parameter `ex_info=True` to the specific logging message.

Example: `logging.error('Could not open file', exc_info=True)`

Lesson 21. Useful Things

21.1 Read a String (Speech)

```
1 from os import system
2 system('say Hello World')
3 system('say -f /Users/donaldelger/Desktop/ct.txt')
```

Line 1 Line 2 reads from a file.

To see more, run the manual command for say like this: `[]: man say`

21.2 Compile a LaTeX file

In terminal, navigate to a folder with LaTeX files. The command `$ latexmk -pdf -gg` will compile the files and build a pdf.

In python, this can be done as follows:
`subprocess.run(['latexmk','-pdf', '-gg'])`

Lesson 22. Arithmetic Operators

Modulo operator `%`, This gives the remainder of a division operations. For example $11\%3 = 2$

Floor operator `//`, in division this gives the integer part of an answer when the answer is expressed as number + remaineder. For example $11//3 = 3$

Part II: Standard Modules

Lesson 23. Math

See **python math module reference** for details.

`degrees(x)` : Convert angle x from radians to degrees

`radians(x)` : Convert angle x from degrees to radians

asin(x): Returns the arc sine of x in radians

Lesson 24. Vector Operations

Cross product

```
>>> x = [1, 2, 3]
>>> y = [4, 5, 6]
>>> np.cross(x, y)
array([-3,  6, -3])
```

Part III: Object Oriented Programming

Lesson 25. Big Idea

Control the state (setting of all attributes)

The object has all the variables needed to control the state.

Lesson 26. Inheritance

```
class ImageBuilder(tk.Toplevel):

    def __init__(self):
        super().__init__(win, bg='powder blue')
```

Lesson 27. Setting and Getting Attributes

setattr(*object*, *name*, *value*)

This is the counterpart of `getattr()`. The arguments are an object, a string and an arbitrary value. The string may name an existing attribute or a new attribute. The function assigns the value to the attribute, provided the object allows it. For example, `setattr(x, 'foobar', 123)` is equivalent to `x.foobar = 123`.

Python's **getattr** function is used to fetch an attribute from an object, using a string object instead of an identifier to identify the attribute. In other words, the following two statements are equivalent:

```
value = obj.attribute
value = getattr(obj, "attribute")
```

To set the attributes of an object from a database row, read the database, extract the row and convert it to a series object, then

```
1 dict = series.to_dict()
2 for key in dict:
3     settattr(self, key, dict[key])
```

Part IV: Techniques for Python Programming

Lesson 28. Timing Code: What Runs Faster?

To see if option A or B runs fast, get some data:

```
1 import timeit
2 print(timeit.timeit('1+3', number=500000))
```

Great intro to the timeit module: **ref**

Best intro to timeit: **ref**

Directly Measure the time **ref1**

Lesson 29. Speaking/Talking

On a mac, you can do this:

```
1 import os
2 os.system('say "your timer has finished"')
```

ref

Lesson 30. Docstring

A **docstring** is a string literal that describes a function or class that occurs at the beginning that uses `"""`

forms the `__doc__` special attribute of the object

For the details of docstrings, see **pep-0257**

Lesson 31. Printing to the Default Printer

This **article** explains printing using terminal.

On a mac, the command `lp myfile` will sent `myfile` to the default printer.

The following code prints a file

```
import subprocess
pn = '/Users/donaldelger/Desktop/dog.png'
subprocess.run(['lp', pn])
```

Lesson 32. CSV to List

Convert CSV to list: This code

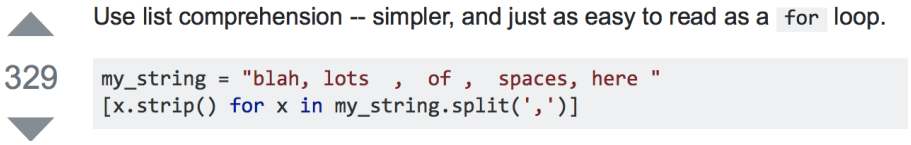
```
str = 'a,b,c'
my_list = str.split(',')
```

will return `['a', 'b', 'c']`

Remove white space about list items:
The method strip() takes a string and removes the white space from the beginning and end of the string. For example:

```
>>> ' The dog runs  a   '.strip()
'The dog runs  a'
```

To get a list and remove white space



Lesson 33. Print to File

See my jupyter notebook file entitled "Print_to_file" in my Learn folder.

```
import sys

orig_stdout = sys.stdout
f = open('/Users/donaldelger/Desktop/dog1.txt', 'w')
sys.stdout = f

for i in range(5):
    print ('i = ', i)

sys.stdout = orig_stdout
f.close()
```

Lesson 34. Regix Searches

A **regular expression** is a sequence of characters that defines a search pattern so that a given string can be searched for the purpose of finding specific parts of this given string.

34.1 Resources

- This page will test regex's **Regex Tester**; killer; watch the how to video
- Another **regex tester**
- Bernd Klien's Python Course **regular expressions** and **advance regular expressions**
- Google education presents this **Regex tutorial**
- **Regex One Tutorial**. I think this site is well done. I rediscovered this site on September 16, 2018.
- **Python.org's** documentation of regexs. The kinder version is **here**
- **Harrison's tutorial**
- **Tutorial**

  Tutorials Point

34.2 MWE

```
import re

string = 'Don was born on Oct 17, 1955. He is 61 years old.'

# This will match (any group of letters) space (any group of numbers)
regex = r"([a-zA-Z]+ \d+)"

# Find the first match; use the group() method to display the match
match_object = re.search(regex, string)
first_match = match_object.group()
print(first_match)

# Find all matches
match_objects = re.findall(regex, string)
print(match_objects)
```

The output is

```
Oct 17
['Oct 17', 'is 61']
```

34.3 The search Function

`re.search(regex, string, flags=0)`

Parameter	Description
regex	The regular expression to be matched
string	The string to be searched
flags	Options to control how the matching is done

34.4 The findall Method

The method `findall()` will find all matches and return a list.

```
import re # Regular expression module

exampleString = '''
Jessica is 15 years old, and Daniel is 27 years old.
Edward is 97 years old, and his grandfather, Oscar, is 102.
'''

ages = re.findall(r'\d{1,3}',exampleString)
names = re.findall(r'[A-Z][a-z]*',exampleString)

print(ages)
print(names)

'''  ********** Output **************
['15', '27', '97', '102']
['Jessica', 'Daniel', 'Edward', 'Oscar']
'''
```

Notes:
`\d{1,3}` means to match all numbers that have a length of 1, 2, or 3.
`[A-Z][a-z]*` means to match all strings that start with a capital letter followed by any number of lower case letters.

34.5 Example

```
mo = re.search('(%fs)(.*)(%fe)', text, re.DOTALL)
```

This example shows

- How to combine the Regex object building with the searching
- How to group into three groups. This allows a string to be pulled out.
- The use of `(.*)` to find all characters (except the new line character).
- The use `re.DOTALL` to match every character including the new line character.

34.6 Methods for regix objects

Performing Matches

Once you have an object representing a compiled regular expression, what do you do with it? Pattern objects have several methods and attributes. Only the most significant ones will be covered here; consult the `re` docs for a complete listing.

Method/Attribute	Purpose
`match()`	Determine if the RE matches at the beginning of the string.
`search()`	Scan through a string, looking for any location where this RE matches.
`findall()`	Find all substrings where the RE matches, and returns them as a list.
`finditer()`	Find all substrings where the RE matches, and returns them as an iterator.

`match()` and `search()` return `None` if no match can be found. If they're successful, a match object instance is returned, containing information about the match: where it starts and ends, the substring it matched, and more.

Figure 34.1: Some methods from the re module

34.7 Group Extraction

match.group() is the whole match text as usual
match.group(1) is the match text corresponding to the 1st left parenthesis
match.group(2) is the text corresponding to the 2nd left parenthesis

34.8 Named Group (Placeholder)

The code that follows is from **ref**. The "P" stands for placeholder.

```
>>> import re
>>> match = re.search('(?P<name>.*) (?P<phone>.*)', 'John 123456')
>>> match.group('name')
'John'
```

34.9 Greedy versus NonGreedy

append a ? after the characters for nongreedy
example: `regex = r'\FRAME.*?special{.*?}}'`

34.10 Tex Doc: Extract text between begin and end

Read the text file. The regex is:
`regex = '(\\\\begin{document}?)(.*)(\\\\end{document}?)'`

The code for extracting the body is:

```
mo = re.search(regex, tex, re.DOTALL)
print('2: ', mo.group(2))
```

34.11 Summary Tables

Regular Expression Basics	
.	Any character except newline
a	The character a
ab	The string ab
a|b	a or b
a*	0 or more a's
\	Escapes a special character

Regular Expression Quantifiers	
*	0 or more
+	1 or more
?	0 or 1
{2}	Exactly 2
{2, 5}	Between 2 and 5
{2,}	2 or more
{,5}	Up to 5

Default is greedy. Append ? for reluctant.

Regular Expression Groups	
(...)	Capturing group
(?P<Y>...)	Capturing group named Y
(?:...)	Non-capturing group
\Y	Match the Y'th captured group
(?P=Y)	Match the named group Y
(?#...)	Comment

Regular Expression Character Classes	
[ab-d]	One character of: a, b, c, d
[^ab-d]	One character except: a, b, c, d
[\b]	Backspace character
\d	One digit
\D	One non-digit
\s	One whitespace
\S	One non-whitespace
\w	One word character
\W	One non-word character

Regular Expression Assertions	
^	Start of string
\A	Start of string, ignores m flag
$	End of string
\Z	End of string, ignores m flag
\b	Word boundary
\B	Non-word boundary
(?=...)	Positive lookahead
(?!...)	Negative lookahead
(?<=...)	Positive lookbehind
(?<!...)	Negative lookbehind
(?()|)	Conditional

Regular Expression Flags	
i	Ignore case
m	^ and $ match start and end of line
s	. matches newline as well
x	Allow spaces and comments
L	Locale character classes
u	Unicode character classes
(?iLmsux)	Set flags within regex

Regular Expression Special Characters	
\n	Newline
\r	Carriage return
\t	Tab
\YYY	Octal character YYY
\xYY	Hexadecimal character YY

Regular Expression Replacement	
\g<0>	Insert entire match
\g<Y>	Insert match Y (name or number)
\Y	Insert group numbered Y

Pattern	Description
^	Matches beginning of line.
$	Matches end of line.
.	Matches any single character except newline. Using m option allows it to match newline as well.
[...]	Matches any single character in brackets.
[^...]	Matches any single character not in brackets
re*	Matches 0 or more occurrences of preceding expression.
re+	Matches 1 or more occurrence of preceding expression.
re?	Matches 0 or 1 occurrence of preceding expression.
re{ n}	Matches exactly n number of occurrences of preceding expression.
re{ n,}	Matches n or more occurrences of preceding expression.
re{ n, m}	Matches at least n and at most m occurrences of preceding expression.
a| b	Matches either a or b.
(re)	Groups regular expressions and remembers matched text.
(?imx)	Temporarily toggles on i, m, or x options within a regular expression. If in parentheses, only that area is affected.
(?-imx)	Temporarily toggles off i, m, or x options within a regular expression. If in parentheses, only that area is affected.
(?: re)	Groups regular expressions without remembering matched text.
(?imx: re)	Temporarily toggles on i, m, or x options within parentheses.
(?-imx: re)	Temporarily toggles off i, m, or x options within parentheses.
(?#...)	Comment.
(?= re)	Specifies position using a pattern. Doesn't have a range.
(?! re)	Specifies position using pattern negation. Doesn't have a range.
(?> re)	Matches independent pattern without backtracking.
\w	Matches word characters.
\W	Matches nonword characters.

Lesson 35. Binary Files

You can read, write, and store a binary file in a variable.

```
1 file_in = '/Users/donaldelger/Desktop/t1.jpg'
2 file_out = '/Users/donaldelger/Desktop/t2.jpg'
3
4 with open(file_in, "rb") as in_file:
5     with open(file_out, "wb") as out_file:
6         file_data = in_file.read()
7         out_file.write(file_data)
```

In line 4, `"rb"` stands for read as binary.

Lesson 36. Raising Exceptions

Use the most specific exception constructor (see the **exception heirachy**) that semantically fits your issue. Be specific in your message. Example:
`raise ValueError('A very specific bad thing happened')`

This **article** explains how to manually raise/throw an exception.

Lesson 37. Pretty Print

For data structures, especially nested dictionaries.

```
1 import pprint
2 pprint.pprint(whateverdatastructureyouwant)
```

```
import pprint
ds = dict((chr(i), list(range(i, i+5))) \
     for i in range(65,70))
pprint.pprint(ds, width=10)
```

Lesson 38. Debugging

```
import pdb; pdf.set_trace()
```

Lesson 39. Subprocess: Running Other Programs

To run illustrator, or other programs, use subprocess:

```
# ==== open a file in illustrator ============================
file = ('/Users/donaldelger/SpyderProjects/Projects/'
        '2017-06_image_editor/figs/joshna.jpg')
app = '/Applications/Adobe Illustrator CS5.1/Adobe Illustrator.app'
subprocess.run(['open', '-a', app, file])
```

This code is in:

```
/Users/donaldelger/SpyderProjects/
Projects/2017-06_image_editor/image_editor.py
```

To compile a LaTeX document, direct the outcome as follows. This keeps a long stream of text outcome from showing up in iPython.

```
try
    subprocess.run(['latexmk', '-f', '-pdf', '-gg', '-shell-escape'],
                   stdout=subprocess.DEVNULL,
                   stderr=subprocess.DEVNULL, check=True)
except subprocess.CalledProcessError:
    log.error(f'Could not compile LaTex File for task {pk}')
    return
```

Part V: Updates and Ecosystem Programs: Terminal, Anaconda, ...

Lesson 40. Updates to Python

install off of the python installation page. then, invoke python in terminal by typing python 3

Lesson 41. Terminal

41.1 Open a File (nondefault app)

```
$ open -a safari  /Users/donaldelger/Desktop/my_image.jpg
```

This does not work for illustrator because the file path has spaces in the name. To get around the spaces, quote the path as follow, or escape the spaces with a `\` or drag and drop the file onto the terminal window.

```
open -a '/Applications/Adobe Illustrator CS5.1/Adobe Illustrator.app' /Users/donaldelger/Desktop/my_im
```

41.2 Run a Function

Lesson 42. Anaconda Updates

```
bin/conda install -c anaconda python=3.6.1
```

```
bin/conda update anaconda
```

```
bin/conda update conda
```

Lesson 43. Spyder

To see the functions, classes, etc (outline): View > Pane > Outline

To update spyder

1. In terminal, navigate to the anaconda folder
2. `bin/conda update spyder`

For updating spyder in a virtual environment:

> To expand on juanpa.arrivillaga's comment:
>
> If you want to update Spyder in the root environment, then `conda update spyder` works for me.
>
> If you want to update Spyder for a virtual environment you have created (e.g., for a different version of Python), then `conda update -n $ENV_NAME spyder` where `$ENV_NAME` is your environment name.

43.1 Run from Spotlight

To run a program created from Spyder using spotlight, open Script Editor and write an AppleScript.

```
tell application "Terminal"
  activate
  do script "python /Users/donaldelger/SpyderProjects/CourseBuilderR1/task_set.py"
end tell
```

Listing 43.1: Example: Running a python program using AppleScript

Lesson 44. Atom IDE

This **video** explains how to setup atom.

Editor: Set tab length to 4 spaces. Soft tabs. Show indent guide.
Packages: Turn on auto saved. Install the package atom-runner.

Open config and add two lines:

```
  runner:
    python: "/usr/local/bin/python3"
```

Use control R to run a script.

Lesson 45. Ipython

45.1 Reset Variables

The command `%reset` resets variables in iPython.

45.2 Run a Script

To run the script `hoshi.py` on the desktop, set the current working directory to the desktop and give ipython the command `In [11]: %run ./hoshi.py`

Lesson 46. Add a module

To install the module tabulate the steps are:

1. run terminal
2. navigate to the folder that holds `\bin`
3. give terminal the command:
   `conda install tabulate`

Lesson 47. jupyter and jupyterlab

47.1 Jupyter

Corey Schaeffer explain how to use jupyter in this **tutorial**.

How to launch Jupyter

1. In terminal, navigate to folder holding notebooks.
2. `$jupyter notebook`

Note: does not seem to work if I leave the canopy virtual environment.

a. Images

Problems: Going up a directory. Absolute image address. Summary to date:

```
<img src="test.png",width=400> ## Works; size scaling
<img src="figs/test.png",width=400> ## Recommended option
<img src="../_figs/test.png",width=400> ## Does not work
![rat](test.png) ## Works; no size scaling
```

47.2 Jupyter Lab

jupyterlab is the next generation of jupyter.

In terminal, give the command `jupyter lab`.

Toggle comments on and off: command/

tutorial

2

1

Part VI: Numpy

For WIP, go to
`/Users/donaldelger/Documents/*A_C/BookCourses/Python/_wip/numpy.md`

This article presents a quick start tutorial for numpy.

`a2 = np.array([])`	create an ndarray from list
`a2.shape`	find dimensions
`a2.reshape(n,m)`	reshape to dimensions $n \times m$
`np.array([1.1]*4)`	creates `([ 1.1, 1.1, ...])`
`np.zeros(4)`	creates [0., 0., 0., 0.])
`np.ones(2)`	creates `([ 1., 1.])`
`np.array(range(0,4))`	creates `([0, 1, 2, 3])`

Lesson 48. Create ndarray

48.1 Passing Lists

```
np.array([[17, 19, 2], [11, 5, 6]], float)
```

creates the following 3×2 array

```
array([[ 17.,  19.,   2.],
       [ 11.,   5.,   6.]])
```

48.2 Random Numbers

The command `np.random.rand(3,2)` creates

```
array([[ 0.50691272,  0.43264325],
       [ 0.99467586,  0.60813541],
       [ 0.6998633 ,  0.6262622 ]])
```

Lesson 49. Basic Math

`np.pi`

`np.sin(theta)`

`np.radians(theta)` : Convert from degrees to radians

`np.linalg.norm(m)` Magnitude of vector m

`np.cross(r, f)` Cross product $\mathbf{r} \times \mathbf{f}$

add vectors; for example add (2, 9, 4) to (1.1, -1, 3)

```
import numpy as np
a, b = (2, 9, 4), (1.1, -1, 3)
c, d = np.array(a), np.array(b)
sum = c + d; print(sum)
# Answer: [3.1, 8., 7.]
```

Lesson 50. Element Wise Operations

Arithmetic operators on arrays always apply to the elements of the array. This is called element wise operations.

```
>>> a, b
(array([ 2.1,  3.4]), array([10, 10]))
>>> a * b
array([ 21.,  34.])
```

Lesson 51. Solving Linear Equations

To solve the equations

$$3x + y = 9x + 2y = 8$$

apply the following code

```
import numpy as np
a = np.array([[3,1], [1,2]])
b = np.array([9,8])
x = np.linalg.solve(a, b)
print(x)
```

to give $x = 2$ and $y = 3$.

Lesson 52. Converting types of data

`tuple(ndarray)` : Convert an ndarray to a tuple

`ndarray.astype(int)` : Convert an ndarray to integer valued

Lesson 53. Random Sample

numpy.random.choice(*a, size=None, replace=True, p=None***)**

Generates a random sample from a given 1-D array

> *New in version 1.7.0.*

Parameters:

a *: 1-D array-like or int*
If an ndarray, a random sample is generated from its elements. If an int, the random sample is generated as if a was np.arange(n)

size *: int or tuple of ints, optional*
Output shape. If the given shape is, e.g., `(m, n, k)`, then `m * n * k` samples are drawn. Default is None, in which case a single value is returned.

replace *: boolean, optional*
Whether the sample is with or without replacement

p *: 1-D array-like, optional*
The probabilities associated with each entry in a. If not given the sample assumes a uniform distribution over all entries in a.

Returns:

samples *: 1-D ndarray, shape (size,)*
The generated random samples

Raises:

ValueError
If a is an int and less than zero, if a or p are not 1-dimensional, if a is an array-like of size 0, if p is not a vector of probabilities, if a and p have different lengths, or if replace=False and the sample size is greater than the population size

See also:
randint, shuffle, permutation

Part VII: Symbolic Math: SymPy

Lesson 54. Misc

54.1 Declaring Symbols

Symbols need to be declared as shown in the example that follows.

```
>>> from sympy import *
>>> x = Symbol('x')
>>> y = Symbol('y')
```

54.2 Integration (Indefinite)

`integrate(f, x)` returns the indefinite integral $\int f dx$

To do the integral $\int kx dx$, the code is

```
from sympy import Integral, Symbol
x = Symbol('x')
k = Symbol('k')
print(Integral(k*x, x).doit())
#  result: k*x**2/2
```

As shown, the code returns $kx^2/2$

54.3 Integration (Definite)

The command `integrate(f, (x, a, b))` evaluates the definite integral $\int_a^b f(x)dx$.

For example, $\int_0^1 0.5(x - x^6)dx$ can be evaluated using

```
from sympy import *
x = Symbol('x')
f = (x-x**6)/2
print(integrate(f,(x,0,1)))
#  Answer: 5/28
```

The code shows that $\int_0^1 0.5(x - x^6)dx = 5/28$

54.4 Integration (Pretty Printing)

For the integral $\int \mathrm{e}^x \cos(x)$, the following commands

```
>>> init_printing(use_unicode=False,
        wrap_line=False, no_global=True)
>>> integrate(exp(x) * cos(x), x)
```

give this formatted result.

$$\frac{\mathrm{e}^x}{2}\sin(x) + \frac{\mathrm{e}^x}{2}\cos(x)$$

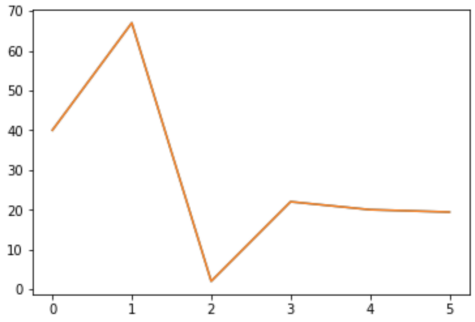
Part VIII: Plotting Data

Lesson 55. Getting Started

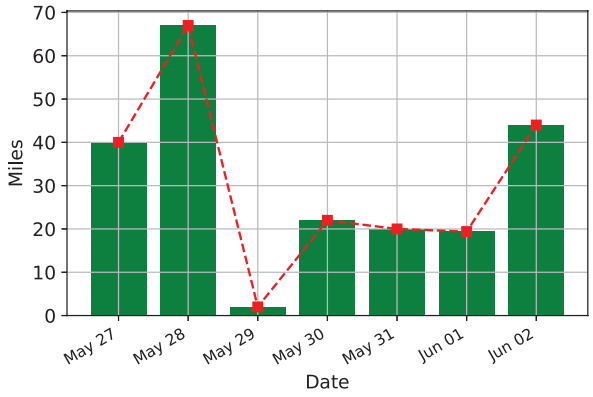
This code, a minimum worked example,

```
import matplotlib.pyplot as plt
plt.plot([40, 67, 2, 22, 20, 19.4])
plt.show()
```

produced this plot



This plot



was produced with this code:

```
def plot_bars(self):
```

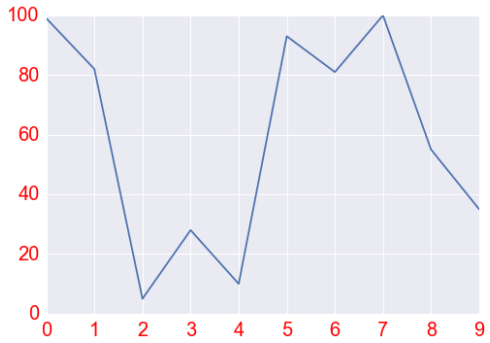
```
    self.get_data()
    self.convert_dates()
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b %d'))
    plt.gca().xaxis.set_major_locator(mdates.DayLocator())
    plt.bar(self.dtobj, self.miles, alpha=0.3, color='green')
    plt.gcf().autofmt_xdate()
    plt.plot(self.dtobj, self.miles,'r--s', alpha=.5)
    plt.tick_params(axis='y', labelsize=12)
    plt.grid(alpha=.4)
    plt.xlabel('Date', fontsize=12)
    plt.ylabel('Miles', fontsize=12)
    plt.savefig('/Users/donaldelger/Desktop/myMileage.eps', format='eps')
    plt.show()
```

Lesson 56. About Plotting

The **seaborn** package provides a way to make publication quality graphics. This **Chris Albon** post shows shows some common statistical plots.

Lesson 57. Style Ticks and Tick Labels

Example: The image that follows shows an increase in the font size and a change in the color.



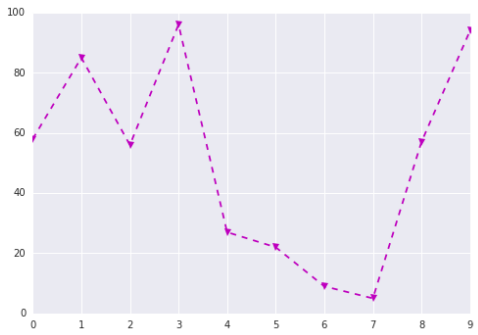
The code to specify the changes is

```
plt.tick_params(axis='both', labelsize=20,
                labelcolor='red')!
```

The method is `.tick_params(axis='both', **kwargs)`. **Matplotlib.org** gives the details.

Lesson 58. Line Types, Colors, and Markers

The image below shows a line that has been modified



This line was modified with the command
`plt.plot(x, y, 'v--m')`

The technique is to pass information with the string that is passed as the third argument. This string, which can be in any order, passes information about the color, the marker, and the line type.

Table 58.1: Line Types

Symbol	Explanation
`'--'`	dashed line
`'-'`	solid line
`':'`	dotted line
`'-.'`	dash dot line

Table 58.2: Line Markers

Symbol	Explanation
`'v'`	triangle 1 down
`'o'`	solid circle
`'^'`	triangle 1 up
`'3'`	triangle 2 left
`'+'`	plus
`'s'`	square

Lesson 59. Saving a Plot

The method is `.savefig(fname, **kwargs)`

The **maplotlib API** gives the details.

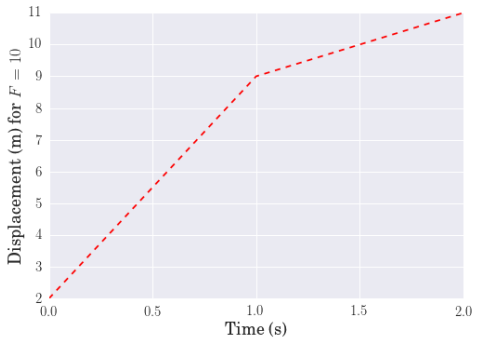
The savefig method needs to come before the `plt.show()` command.

```
fn = '/Users/donaldelger/Desktop/123_vp2.pdf'
plt.savefig(fn, format='pdf')
plt.show()
```

Listing 59.1: An example of saving a file

Lesson 60. LaTeX Labels

The figure that follows shows an example



This example was created using

```
1  import seaborn
2  import matplotlib.pyplot as plt
3  plt.rc('text', usetex=True)
4  plt.rc('font', family='serif')
5  plt.plot([2, 9, 11], '--r')
6  plt.xlabel(r'Time (s)', fontsize=18)
7  plt.ylabel(r'Displacement (m) for $ F = 10 $',
8             fontsize=18)
9  plt.tick_params(axis='both', labelsize=15)
10 plt.show()
```

Lines 3 and 4 gives the rc settings needed to invoke LaTeX. Regarding "rc", a **stackoverflow post** states that configure files are often ended in "rc", e.g., `.xinitrc`. Configurations *r*un and they *c*onfigure your stuff. This practice started before unix.

This **matlplotlib** webpage gives details about using LaTeX.

Part IX: SQLite

Lesson 61. Minimum Program

```
import sqlite3
db = 'test.db'
conn = sqlite3.connect(db)
c = conn.cursor()
c.close()
conn.close()
```

Lesson 62. Select a Record

```
query = 'SELECT * FROM figs WHERE pk={}'.format(pk)
data = c.execute(query)
values = data.fetchall()[0]
```

Lesson 63. Insert a record

Follow this structure!

```
new = 'moose', 5, 2.11
c.execute('INSERT INTO main (a, b, c) VALUES (?, ?, ?)', new)
conn.commit()
```

Insert an empty record:
`c.execute('INSERT INTO figs DEFAULT VALUES'`

Lesson 64. Find the Maximum of a Column

```
bb = c.execute('SELECT MAX (goals3_pk) FROM goals3')
pk = bb.fetchall()[0][0] + 1
```

Lesson 65. Delete a Row

parameterized query

```
self.c.execute('DELETE FROM goals3 WHERE
    goals3_pk = ?', (pk,))
```

Lesson 66. Get Column Names

```
data = c.execute("PRAGMA table_info(figs)")
d2 = data.fetchall()
col_names = [d2[i][1] for i in range(len(d2))]
```

Note: The SQLite PRAGMA command can be used to read or set various environmental variables and state flags

Lesson 67. Get Number of Records and Last Record

```
query = 'SELECT Count(*) FROM tasks'
data = c.execute(query)
print('Number of tasks: {}'.format(data.fetchall()[0][0]))
```

Listing 67.1: Number of records

```
c.execute("SELECT * FROM tasks ORDER BY pk DESC LIMIT 1")
result = c.fetchone()
last_pk = result[0]
print('last_pk: {}'.format(last_pk))
```

Listing 67.2: Last Record

Lesson 68. Get ID of Last Row Inserted

get the lastrowid attribute from the cursor object

```
self.pk = c.lastrowid
```

Lesson 69. Insert Date or Timestamp as Default

In the db browser, when you create or modify the table insert CURRENT_DATE in the default field.

ref

Lesson 70. Parameterized Query

Key Idea: Do not assemble a query using Python's string operations because doing so makes your program vulnerable to an SQL injection attack.

Ref.

A PQ is a "variable" or "parameter" token that specifies a placeholder in the expression for a value that is filled in at runtime using the sqlite3_bind() family of C/C++ interfaces.

Example 1. Notice the comma:

```
c.execute('DELETE FROM goals WHERE pk = ?', (pk,))
```

Example 2:

```
c.execute('UPDATE goals SET desc=? WHERE pk=?', (new, pk))
```

Example 3:

```
purchases = [('2006-03-28', 'BUY', 'IBM', 1000, 45.00),
             ('2006-04-05', 'BUY', 'MSFT', 1000, 72.00),
             ('2006-04-06', 'SELL', 'IBM', 500, 53.00),
            ]
c.executemany('INSERT INTO stocks VALUES (?,?,?,?,?)', purchases)
```

Lesson 71. Row Dictionary

The command `df.to_dict(orient='records')` reads each row of a dataframe into a dictionary and stores the collection of dictionaries to a list. The code below reads one row into a df and then extracts this row into a dictionary.

```
query = 'SELECT * FROM ats WHERE pk={}'.format(pk)
df = pd.read_sql_query(query, conn)
record = df.to_dict(orient='records')[0]
```

Lesson 72. Read Table to Pandas

```
df = pd.read_sql_query(
    "SELECT * FROM main ORDER BY seq", conn)
```

Lesson 73. Read Table, Modify, Write to sqlite3

```
import pandas as pd
import sqlite3
db = '/Users/donaldelger/Desktop/build_goals/goals.db'
conn = sqlite3.connect(db)
c = conn.cursor()
df = pd.read_sql_query("SELECT * FROM g1 ORDER BY seq", conn)
print(df)
df.iloc[1,1] = 'Smile often'
print(df)
c.executescript('drop table if exists g1;')
df.to_sql('g1', conn)
```

Lesson 74. Create New Table or DataBase

To create a new dbase, use the connect method and this will automatically create a new file.

The main idea is to create a table. This example

```
        try:
            conn = sqlite3.connect(self.dbase)
            c = conn.cursor()
            qy = """
                  CREATE TABLE goals (
                  pk    INT PRIMARY KEY     NOT NULL,
                  desc  TEXT                NOT NULL,
                  path  TEXT,
                  mpath TEXT
                  )
                  """
            c.execute(qy)
            logging.info('new course dbase successfully created')
        except:
            logging.error('Could not create a new dbase for the course\n')
```

shows the creation of a simple table. Notice that a primary key will auto increment by default. Thus, I did not specify a keyword.

Lesson 75. Sqlite Shell

Ref

sqlite3 or sqlite3 db path: (to start)
terminate commands with ;
dot commands: .databases

75.1 Copying a Table from one dbase to another

ref

```
insert into main.gg select * from AM.gtest;
attach '/Users/donaldelger/Desktop/sqlite_learn/sq1.db' as AM;
```

Part X: Pandas

Lesson 76. Series

76.1 Series to list

```
weight = pd.Series.tolist(criteria_df['weight'])
```

Lesson 77. Create a DataFrame

killer article

77.1 From Records (rows)

```
     owner  dogs
0      ben     0
1  michael     1
2    linda     1
```

```
import pandas as pd
columns = ['owner', 'dogs']
rows=[('ben', 0), ('michael',1), ('linda', 1)]
df = pd.DataFrame.from_records(rows, columns=columns)
print(df)
```

77.2 From numpy ndarray

a. Unlabeled rows and columns

The following dataframe

```
   0                      1    2
0  2  Build course builder  0.9
1  7   Build three courses  1.1
```

was created using

```
import pandas as pd
import numpy as np
data = np.array([
    [2, 'Build course builder', .9],
    [7, 'Build three courses', 1.1]])
df = pd.DataFrame(data)
```

b. Labeled Rows and Columns

The dataframe

```
       pk  task                    seq
---  ----  --------------------  -----
pk2     2  Build course builder    0.9
pk1     7  Build three courses     1.1
```

was created using

```
import pandas as pd
import numpy as np
import tabulate
data = np.array([
    [2, 'Build course builder', .9],
    [7, 'Build three courses', 1.1]])
df = pd.DataFrame(data, columns=
        ['pk','task','seq'], index=['pk2','pk1'])
print(tabulate.tabulate(df,
      headers=list(df.columns)))
```

77.3 From dictionary

```
d = {'a': [1, 2, 3], 'b': [4, 5, 6]}  # dict
df = pd.DataFrame(d)

""" Result
   a  b
0  1  4
1  2  5
2  3  6 """
```

77.4 From Excel File

```
import pandas as pd

fname = '/Users/donaldelger/Desktop/test.xlsx'

df = pd.read_excel(fname, sheetname=0)
# can also index sheet by name or fetch all sheets
mylist = df['pk'].tolist()
```

77.5 From sqlite3 DataBase

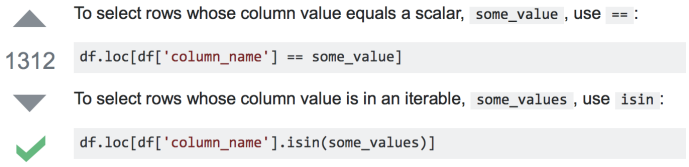
```
import sqlite3
import pandas as pd
con = sqlite3.connect('winco.db')
df = pd.read_sql_query('SELECT * FROM items ORDER by location_fk', con)
```

Lesson 78. Cells

Get value from a cell:
`df.get_value('row', 'cell')`

Set a value of a cell:
`df.set_value('row', 'cell', 'new ...')`

78.1 Given a cell value, Get the row



Lesson 79. Column Operations

79.1 Get One or More Columns

`df['seq']`

`df[['seq', 'goal']]`

79.2 Column to List

`categories = df['categories'].to_list()`

79.3 List Column Names

```
list(df.columns)
df.columns.tolist()
```

79.4 Get the Index

```
records = df.index.values
```

`df.index`: this will return an iterable

`df.index.tolist()` this will return a list

records is a numpy ndarray object (iterable)

79.5 Sum a column

`total = df['MyColumn'].sum()`

79.6 Divide values in one column by another column

```
df['c'] = df['b']/df['a']
```

79.7 Delete a Column

The command is `df.drop('seq', axis=1)`. The `axis=1` parameter denotes that a column (not a row) is being dropped.

79.8 Add a Column

Use the command `df['seq'] = seq2`, where `'seq'` is the name of the new column

79.9 Sort a df by One Column

To sort by column named `seq`, use `df.sort_values('seq', inplace=True)`

79.10 Specify The Order in Which the Columns Appear

```
df.reindex_axis([list of column names sorted in desired order], axis=1)
```

Lesson 80. Number of Rows/Columns in a df

```
len(df)  # number rows
len(df.columns) # number of columns
df.shape # number of rows and columns
Out[1]: (6, 9)
```

Lesson 81. Work with Rows

81.1 Iterate over Rows in a Dataframe

```
for (index, row) in df.iterrows():
```

81.2 Add a row to a DataFrame

Step 1: Create a dataframe with one row

Step 2: Append the new dataframe to the existing df

81.3 Select Certain Rows

Select all rows with `'level'` equal to 1.

```
df[df['level'] == 1]
```

81.4 Select One Row; More that one row

```
df[0:1] # select row with index 0
df[3:4] # select row with index 3
```

Select a slice of rows by integer position
[inclusive-from : exclusive-to [: step]]
default start is 0; default end is len(df)

```
df = df[:]       # copy DataFrame
df = df[0:2]     # rows 0 and 1
df = df[-1:]     # the last row
df = df[2:3]     # row 2 (the third row)
df = df[:-1]     # all but the last row
df = df[::2]     # every 2nd row (0 2 ..)
```

Trap: a single integer without a colon is a column label for integer numbered columns.

81.5 Select Row(s) based on Partial String Search

```
df[df['text'].str.contains("flu")]
```

81.6 Select Rows based on a List

This `ref` gives a simple example.

```
mylist = ['granola', 'salami']
df2 = df[df['item'].isin(mylist)]
```

81.7 Reset the Index

Slice some rows out and then reindex to 0, 1, 2, ...

```
df = df.reset_index(drop=True)
```

Need the drop=True or an extra column "index" is added

Lesson 82. Convert a Series to a Dictionary

```
for (index, row) in df.iterrows():
    video_record = row.to_dict()
```

Lesson 83. Deep Copy

The command `df.copy(deep=True)` below is useful sometimes if a shallow copy is an issue.

Lesson 84. Dropping Rows and Columns

drop several rows

```
df.drop(['Cochise', 'Pima'])
```

```
drop(    labels=None,    axis=0,    index=None,
     columns=None,    level=None,
     inplace=False,
     errors='raise',)Docstring:Drop specified
     labels from rows or columns.Remove rows or
     columns by specifying label names and
     correspondingaxis, or by specifying directly
     index or column names. When using amulti-
     index, labels on different levels can be
     removed by specifyingthe level.Parameters
     ----------labels : single label or list-like
     Index or column labels to drop.axis : {0 or
     'index', 1 or 'columns'}, default 0
     Whether to drop labels from the index (0 or
     'index')
```

Drop a row if it contains a certain value (in this case, "Tina")

Specifically: Create a new dataframe called df that includes all rows where the value of a cell in the name column does not equal "Tina"

```
df[df.name != 'Tina']
```

Ref: Chris Albon

Lesson 85. Join Dataframes

```
frames = [df1, df2, df3, ...]
df = pd.concat(frames)
df.reset_index(drop=True)
```

Lesson 86. DataFrame to SQL

pandas.DataFrame.to_sql

`DataFrame.to_sql`(*name*, *con*, *flavor=None*, *schema=None*, *if_exists='fail'*, *index=True*, *index_label=None*, *chunksize=None*, *dtype=None*) [source]

Write records stored in a DataFrame to a SQL database.

Parameters:

name *: string*
Name of SQL table

con *: SQLAlchemy engine or DBAPI2 connection (legacy mode)*
Using SQLAlchemy makes it possible to use any DB supported by that library. If a DBAPI2 object, only sqlite3 is supported.

flavor *: 'sqlite', default None*
Deprecated since version 0.19.0: 'sqlite' is the only supported option if SQLAlchemy is not used.

schema *: string, default None*
Specify the schema (if database flavor supports this). If None, use default schema.

if_exists *: {'fail', 'replace', 'append'}, default 'fail'*
- fail: If table exists, do nothing.
- replace: If table exists, drop it, recreate it, and insert data.
- append: If table exists, insert data. Create if does not exist.

index *: boolean, default True*
Write DataFrame index as a column.

index_label *: string or sequence, default None*
Column label for index column(s). If None is given (default) and *index* is True, then the index names are used. A sequence should be given if the DataFrame uses MultiIndex.

chunksize *: int, default None*
If not None, then rows will be written in batches of this size at a time. If None, all rows will be written at once.

dtype *: dict of column name to SQL type, default None*
Optional specifying the datatype for columns. The SQL type should be a SQLAlchemy type, or a string for sqlite3 fallback connection.

Lesson 87. DataFrame to other formats

87.1 DataFrame to Table

a. Working Example

```
body2 = tabulate(dfx, headers="keys", tablefmt="latex", showindex="never")
```

b. Background

This **website** explains how to use tabulate.

The module **tabulate** converts a pandas dataframe to a table. This module needs to be install into the anaconda environment. The code below provides an example

```
import pandas as pd
import tabulate

pk = [2, 17, 1]
seq = [.5, 1, 4.5]
desc = ['define ct', 'define claim', 'define truth']
dic = {'pk': pk, 'seq': seq, 'desc': desc}
df = pd.DataFrame(data=dic)
col_labels = ['i', 'desc', 'pk', 'seq']
print(tabulate.tabulate(df, headers=col_labels))
```

The output is

```
  i  desc            pk    seq
---  ------------  ----  -----
  0  define ct        2    0.5
  1  define claim    17    1
  2  define truth     1    4.5
```

87.2 DataFrame to csv

For the previous table, the command `df.to_csv(index_label="num")` gives the following string (carriage returns were added:

```
'num,desc,pk,seq
\n0,define ct,2,0.5
\n1,define claim,17,1.0\
n2,define truth,1,4.5\n'
```

Part XI: Tkinter

tkdocs tutorial

Bernd Klein's Tutorial

Lesson 88. Big Picture: How to Use TkInter

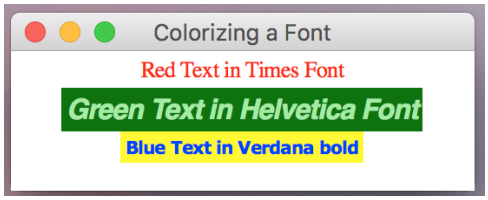
Good Overview

Bryan Oakley on Stack Overflow

```
import tkinter as tk
from tkinter import ttk
class SimpleTable(tk.Frame):
    def __init__(self, win, df):
        super().__init__(master = win)
        ttk.Button(self, text='test').grid()

root = tk.Tk()
mt = SimpleTable(root, df)
root.geometry('300x400')
mt.grid()
root.mainloop()
```

Lesson 89. Label Widget

Ref.

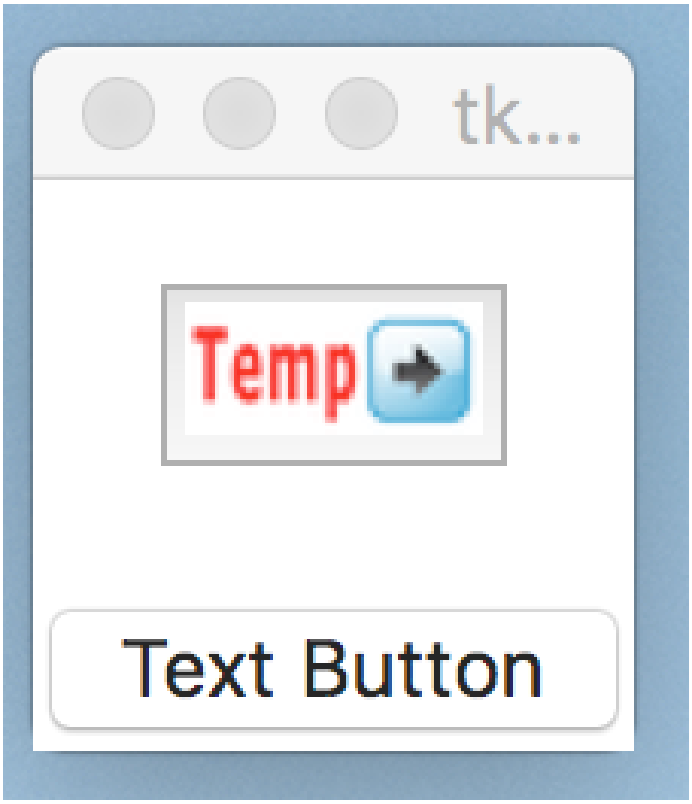



```
from tkinter import *
root = Tk()
root.title("Colorizing a Font")

Label(root,
                  text="Red Text in Times Font",
                  fg = "red",
                  font = "Times").pack()
Label(root,
                  text="Green Text in Helvetica Font",
                  fg = "light green",
                  bg = "dark green",
                  font = "Helvetica 16 bold italic").pack()
Label(root,
                  text="Blue Text in Verdana bold",
                  fg = "blue",
                  bg = "yellow",
                  font = "Verdana 10 bold").pack()

root.mainloop()
```

Lesson 90. Button

90.1 Use Image

The next image shows a button displayed with an image and one displayed with text.

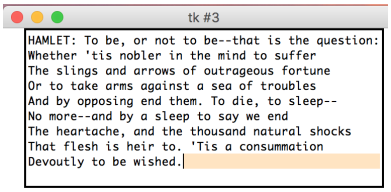



```
import tkinter as tk
from PIL import ImageTk, Image

def talk():
    print('hi there')

root = tk.Tk()
image = Image.open("temp.png")
new_size = int(22 / image.size[1] * image.size[0]), 22  # 22 pixels high
image2 = image.resize(new_size, Image.ANTIALIAS)
image3= ImageTk.PhotoImage(image2)
b = tk.Button(root, text="hello", image=image3, command=talk)
```

```
13 b.image = image
14 b.grid(padx=15, pady=15)
15 tk.Button(root, text='Text Button').grid()
16 root.mainloop()
```

Lesson 91. Text Widget

The window

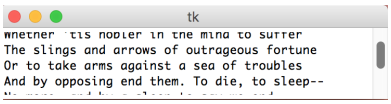


was generated by

```
from tkinter import *
root = Tk()
T = Text(root, height=10, width=50)
T.pack()
quote = """HAMLET: To be, or not to be--
that is the question ... """
T.insert(END, quote)
mainloop()
```

note: use `tk.END` for the index for common module import

The following image shows a scroll bar



that was added using

```
from tkinter import *
root = Tk()
S = Scrollbar(root)
T = Text(root, height=4, width=50)
S.pack(side=RIGHT, fill=Y)
T.pack(side=LEFT, fill=Y)
S.config(command=T.yview)
T.config(yscrollcommand=S.set)
quote = """HAMLET: To be, or not to be ..."""
T.insert(END, quote)
mainloop(  )
```

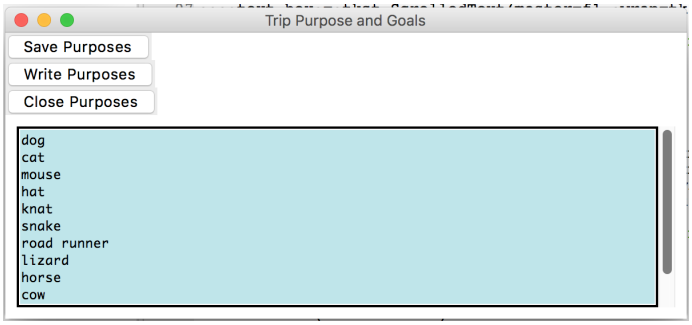
91.1 Get Text

```
box.get('1.0', 'end-1c')
```

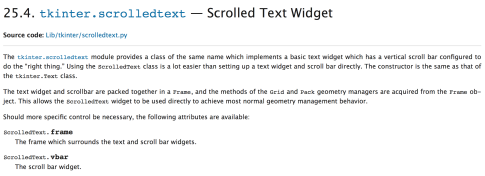
The '1.0' means to read text from line 1 character zero. The 'end-1c' means to read text to the end character minus one character. Without subtracting one character, a line break will be added.

Lesson 92. ScrolledText Widget

Combine a text box and a vertical scrollbar:




```
import tkinter.scrolledtext as tkst
text_box = tkst.ScrolledText(master=w2, wrap=tk.WORD, width=79, height=10, bg='powder blue')
text_box.grid(padx=10, pady=10)
tplate = 'See .. \nExperience ..\nEngage in ..\nDine at ..\Check out'
text_box.insert(tk.INSERT, tplate)
```



25.4. tkinter.scrolledtext — Scrolled Text Widget

Source code: Lib/tkinter/scrolledtext.py

The tkinter.scrolledtext module provides a class of the same name which implements a basic text widget which has a vertical scroll bar configured to do the "right thing." Using the ScrolledText class is a lot easier than setting up a text widget and scroll bar directly. The constructor is the same as that of the tkinter.Text class.

The text widget and scrollbar are packed together in a Frame, and the methods of the Grid and Pack geometry managers are acquired from the Frame object. This allows the ScrolledText widget to be used directly to achieve most normal geometry management behavior.

Should more specific control be necessary, the following attributes are available:

ScrolledText.frame
The frame which surrounds the text and scroll bar widgets.

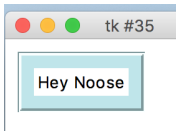
ScrolledText.vbar
The scroll bar widget.

To get the contents of the window: example

```
scrolltext_widget_name.get(1.0, tk.END)
```

start at line 1, character zero and finish at the end of the text; returns a string

Lesson 93. Frame



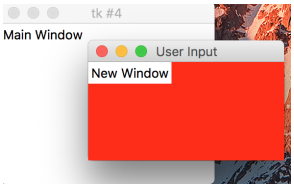

```
import tkinter as tk
root = tk.Tk()
root.geometry('200x200+300+300')
f1 = tk.Frame(root, bd=3, relief='groove', bg='powder blue')
f1.grid(padx=10, pady=10)
tk.Label(f1, text="Hey Noose").grid(padx=10, pady=10)
root.mainloop()
```

Lesson 94. LabelFrame

```
f1 = tk.LabelFrame(self, bd = 2 , text = "Learning Outcomes",
                   fg = 'red', relief = 'raised', padx = 5,
                   pady = 5, font = "Verdana 12 bold")
```

Lesson 95. Toplevel

The toplevel window



was created using

```
import tkinter as tk
root = tk.Tk()
root.geometry('220x200')
tk.Label(root, text="Main Window").grid()
top = Toplevel(bg='red')
top.title("User Input")
top.geometry('200x100')
top.lift(root)
tk.Label(top, text="New Window").grid()
root.mainloop()
```

Lesson 96. Entry Widget

Fig. 96.1 shows an example of an entry widget

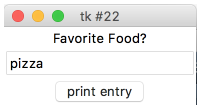


Figure 96.1: An entry widget

The code to create Fig. 96.1 is presented in Listing 96.

```
1  import tkinter as tk
2
3
4  def pt():
5      print(dd.get())
6
7  root = tk.Tk()
8  tk.Label(root, text="Favorite Food?").grid()
9  dd = tk.Entry(root)
10 dd.grid()
11 dd.insert(0, 'pizza')
12 tk.Button(root, text='print entry', command=pt).grid()
13 root.mainloop()
```

Listing 96.1: Listing for Entry Widget Code

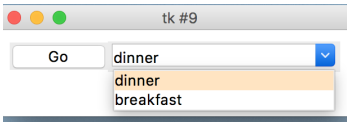
96.1 Updating an Entry Box

```
1
2  # set up the entry box
3  pk_box = tk.Entry(self)
4  pk_box.grid()
5  pk = 22
6  pk_box.insert(0, pk)
7
8  # update the entry box after pk has changed
9  pk_box.delete(0, 'end')
10 new_pk = 29
11 pk__box.insert(0, new_pk)
```

96.2 Scrolling Entry Widget

```
import tkinter as tk
root = tk.Tk()
scrollbar = tk.Scrollbar(orient="horizontal")
e3 =tk.Entry(xscrollcommand=scrollbar.set)
e3.focus()
e3.pack(side="bottom",fill="x")
#e3.grid(row=10, column=7)
scrollbar.pack(fill="x")
scrollbar.config(command=e3.xview)
root.mainloop()
```

Lesson 97. Combobox

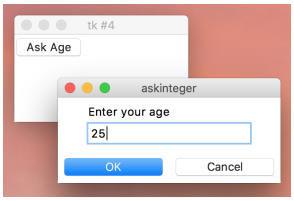



```
#======== Combo Box to Select CookBook Categories ============
cats = self.cookbook.categories()  # List to display
cb = ttk.Combobox(self, values=cats)
cb.grid(row = 0, column=1, sticky = tk.W)
cb.set(cats[0])
self.combobox = cb
```

Lesson 98. simpledialog

98.1 Example

The following window gets user input.



To get the selected value from the combobox, use `self.combogox.get()`

The previous window was created with

```
import tkinter.simpledialog as sd
import tkinter as tk
from tkinter import ttk

def ask_age():
    age = sd.askinteger("askinteger", "Enter your age")
    print(age)

root = tk.Tk()
root.geometry('200x100')
tk.Button(root, text='Ask Age', command=ask_age).grid()
root.mainloop()
```

98.2 Parameters

```
seq = sd.askstring('Seq Number?', 'Sequence number?'
```

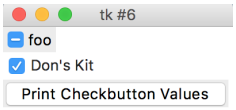
```
chap = sd.askinteger("Print New Problems", "Chapter number?")
```

Lesson 99. Checkbutton Widget

The purpose of a checkbutton widget is to allow the user to read and select a two-way choice.

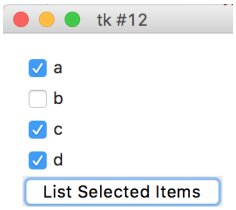
Ref.

Use tk not ttk due to scant ttk documentation




```
import tkinter as tk
from tkinter import ttk

def pack_list():
    print(var.get(), chk.state())

tkwindow = tk.Tk()
tkwindow.geometry('200x200')

# checkbox with ttk
chk = ttk.Checkbutton(tkwindow, text="foo")
chk.grid(column=0, row=0, sticky='W')

# checkbox with tk
var = tk.IntVar()
var.set(1) # how to set default status as checked ...
cb = tk.Checkbutton(tkwindow, text='Don\'s Kit', variable=var)
cb.grid(sticky='W')

ttk.Button(tkwindow, text="Print Checkbutton Values", command=pack_list).grid()

tkwindow.mainloop()
```

99.1 Example



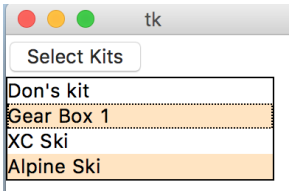
path name: `../Learn/tkCheckbutton/checkbutton_selector.py`

```
import tkinter as tk
from tkinter import ttk

def get_items():
    print()
    for (i, item) in enumerate(items):
        if cb[i].get():
            print(item)

items = ['a', 'b', 'c', 'd']

root = tk.Tk()
f1 = tk.Frame(root)
f1.grid(padx=15, pady=15)

cb_status = [0, 1, 0, 1]
cb = []
for (i, item) in enumerate(items):
    cb.append(tk.IntVar())
    cb[i].set(cb_status[i])
    widget = tk.Checkbutton(f1, text=item, variable=cb[i])
    widget.grid(sticky=tk.W)

ttk.Button(f1, text='List Selected Items', command=get_items).grid(sticky=tk.W)

tk.mainloop()
```

Lesson 100. Listbox Widget

The purpose of a listbox is to display a list and then allow the user to select one or more item.

The **ref** describes how to add a scroll bar to a Listbox widget.



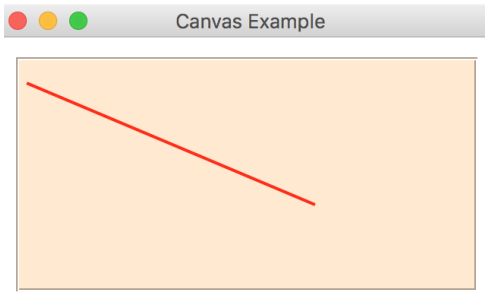

```
# Listbox with tk
import tkinter as tk

def pkits():
    """ print the selected list iterms """
    index = kits_lbox.curselection()
    for i in index:
        print(kits[i])

tkwindow = tk.Tk()
tkwindow.geometry('200x200')

tk.Button(tkwindow, text='Select Kits', command=pkits).grid(sticky=tk.W)

kits = ['Don\'s kit', 'Gear Box 1', 'XC Ski', 'Alpine Ski']

kits_lbox = tk.Listbox(tkwindow, selectmode=tk.MULTIPLE, height=len(kits))
kits_lbox.grid()

for kit in kits:
    kits_lbox.insert(tk.END, kit)

tkwindow.mainloop()
```

Lesson 101. Canvas

Some references are:

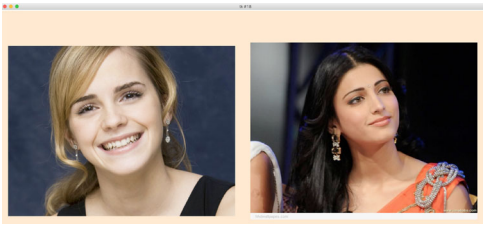
- **tkdocs tutorial**
- Great tutorial: **here**.
- **EU site**

101.1 Line




```
import tkinter as tk
root = tk.Tk()
root.geometry('350x200+1000+0')
root.title('Canvas Example')
w = tk.Canvas(root, borderwidth=2, bg='bisque', relief='groove',
              width=300, height=150)
w.grid(padx=10, pady=10)
w.create_line(10, 20, 200, 100, fill='red', width=2)
root.mainloop()
```

101.2 Image



```
# Add a path to CourseBuilder
import sys
sys.path.append('/Users/donaldelger/SpyderProjects/CourseBuilderR1')

import tkinter as tk
import image

root = tk.Tk()
root.geometry('1600x800+700+0')

# ============= Add the canvas ============================
c1 = tk.Canvas(root, width=1600, height=800, bg='bisque', relief='groove')
c1.grid()

# =========== Add the leftmost image ==============================
image1 = '/Users/donaldelger/SpyderProjects/Learn/tkCanvas/figs/p3.jpg'
i1 = image.Image4tk(image1).tkimg
c1.create_image(400, 400, image=i1)
c1.image = i1  # prevent the tk.image problem by saving the file

# =========== Add the rightmost image ==============================
image2 = '/Users/donaldelger/SpyderProjects/Learn/tkCanvas/figs/p4.jpg'
i2 = image.Image4tk(image2).tkimg
c1.create_image(1200, 400, image=i2)
c1.image = i1

root.mainloop()
```

Lesson 102. Interact with User: Dialog, Message, File

This works:

```
""" Return the full path to the excel file containing task pks """
root = tk.Tk()
root.withdraw()
excel_folder = '/Users/donaldelger/Documents/__EFM12E/Excel_Lists'
filename = askopenfilename(parent=root, initialdir=excel_folder)
root.update()
root.destroy()
return filename
```

The filedialog module. See **this article**.

.askopenfilename(option=value, ...) Select an existing file.

.asksaveasfilename(option=value, ...) Create or modify an existing file.

```
from tkinter import Tk
from tkinter.filedialog import askopenfilename

Tk().withdraw()
filename = askopenfilename()
print(filename)
```

to start with a initial directory:

```
filename = askopenfilename(initialdir=mydir)
```

Code that works for opening an existing file (on desktop)

```
from tkinter import filedialog as fd
import tkinter as tk
root = tk.Tk()
md = '/Users/donaldelger/Desktop'
fn = fd.askopenfilename(initialdir=md)
root.destroy()
print(fn)
```

Problem: file selection window does not close. Solution:

```
    root = tk.Tk()
    md = '/Users/donaldelger/Documents/*A_C/BookCourses/Finance/tasks/4/views'
    filename = askopenfilename(initialdir=md)
    root.destroy()
```

102.1 Get a New File Name From User

```
from tkinter import filedialog as fd
  folder = '/Users/donaldelger/Desktop/gear/'
  if not os.path.isdir(folder):
      os.mkdir(path)
  fn = fd.asksaveasfilename(initialdir=folder, defaultextension='.tex',
                            initialfile='gear_list')
  if fn:
      print(fn)
```

Lesson 103. Open/Create a File or Folder

```
fname=fd.asksaveasfile(initialdir=folder, initialfile='name.tex')
```

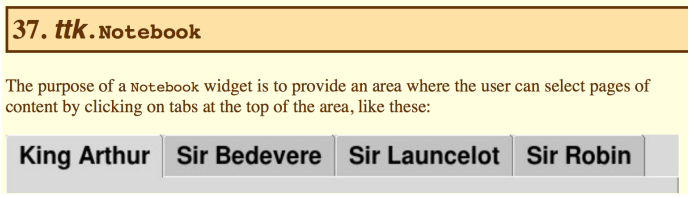
NMT describes the details

103.1 Open/Create a Folder

title does not seem to do anything

```
import tkinter as tk
from tkinter import filedialog as fd
root = tk.Tk()
root.directory = fd.askdirectory(title=my_title, initialdir=my_path)
print(root.directory)
root.mainloop()
```

Lesson 104. Notebook Widget (Tabbed)




```
import tkinter as tk
from tkinter import ttk
from PIL import ImageTk, Image

root = tk.Tk()
n = ttk.Notebook(root)
f1 = ttk.Frame(n)   # first page, which would get widgets gridded into it
f2 = ttk.Frame(n)   # second page
n.add(f1, text='One')
n.add(f2, text='Two')
n.grid()

ttk.Label(f1, text = 'howdy hoshi').grid()
ttk.Label(f2, text = 'walk hoshi').grid()

root.mainloop()
```

Listing 104.1: Example of a simple notebook

Lesson 105. Message Box

105.1 Warning Message Box

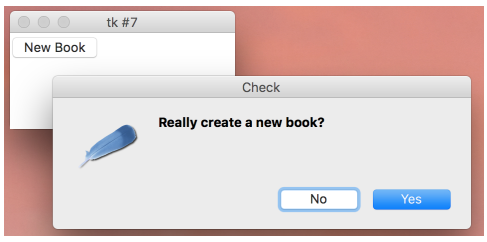
A message box looks like

The code that created this example is

```
import tkinter as tk
from tkinter import messagebox

def test():
    messagebox.showerror("Warning!", "Error in ...")

top = tk.Tk()
top.geometry("200x100")
B1 = tk.Button(top, text="Test MessageBox",
               command=test)
B1.place(x=35, y=50)
top.mainloop()
```

105.2 Yes/No Messagebox

This message box is used for logic.



```
import tkinter as tk
from tkinter import messagebox


def new_book():
    if messagebox.askyesno('Check', 'Really create a new book?'):
        # call NewBook here
        pass
    return

root = tk.Tk()
tk.Button(root, text='New Book', command=new_book).grid()
root.mainloop()
```

The following code asks one yes/no question and then destroys the widget

```
import tkinter as tk
from tkinter import messagebox

class Question:
    """ Ask one question and then destroy the widget """
    def __init__(self):
        self.root = tk.Tk()
        tk.Label(self.root, text='Move Forward? ').grid()
        self.root.geometry('300x50+0+0')
        self.response = messagebox.askyesno('Check',
                                            'Really create new task folders?')
        self.root.destroy()
        self.root.mainloop()

a = Question()
print(a.response)
```

105.3 Box Types

showinfo()
showwarning()
showerror ()
askquestion()
askokcancel()
askyesno ()
askretrycancel ()

Lesson 106. The filedialog module

106.1 Overview of the 3 module functions

Function	Parameters	Purpose
.askopenfilename	Directory, Title, Extension	To **open** file: Dialog that requests selection of an existing file.
.asksaveasfilename	Directory, Title, Extension)	To **save** file: Dialog that requests creation or replacement of a file.
.askdirectory	None	To **open directory**

106.2 askdirectory

You can get an existing directory or create a new directory.

The function returns the name of the directory.

```
import tkinter as tk
from tkinter import ttk
from tkinter import filedialog as fd

def fld():
    dir = '/Users/donaldelger/Desktop'
    directory = fd.askdirectory(initialdir=dir,
                                message='select the root folder')
    print(directory)

root = tk.Tk()
ttk.Button(root, text='Get Folder', command=fld).grid(padx=25, pady=25)
root.mainloop()
```

Lesson 107. Entry box, Get Data on Return

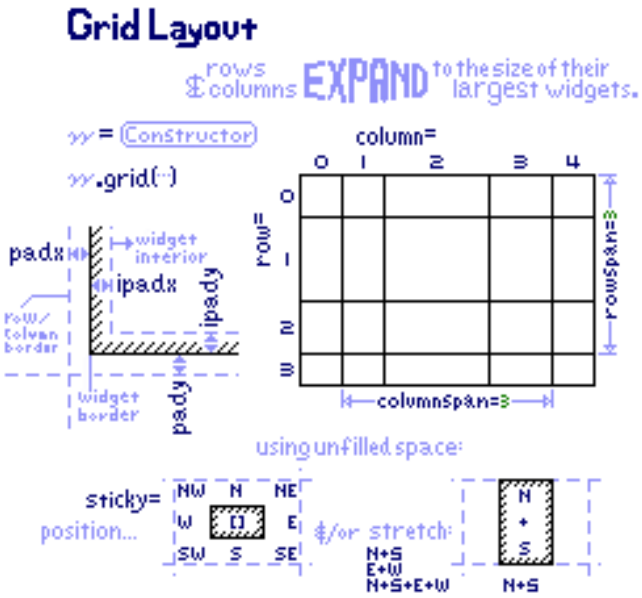
task. create an entry box such that a return will read the data

```
def callback(event):
    print(dd.get())

# Build a Gui and get the cell value on return ...
import tkinter as tk
root = tk.Tk()
root.geometry('500x200+0+1000')
dd = tk.Entry(root)
dd.grid()
dd.bind("<Return>", callback)
root.mainloop()
```

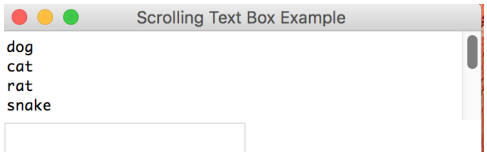
Recover which widget was clicked: `event.widget`

Get widget's data: `event.widget.get()`

Lesson 108. Grid



Lesson 109. Text Box with Scrolling




```
from tkinter import *
root = Tk()
root.title('Scrolling Text Box Example')
S = Scrollbar(root)
T = Text(root, height=4, width=50)
S.grid(sticky='nsew', row=0, column=1)
T.grid(sticky='nsew', row = 0, column=0)
S.config(command=T.yview)
T.config(yscrollcommand=S.set)
petlist = 'dog\ncat\nrat \nsnake\nhampster\nfish \nsnake\nhampster\nfish  '
T.insert(END, petlist)
Entry(root).grid(sticky='W')
mainloop(  )
```

Lesson 110. Passing Variables Using Lambda

A lambda function without parameters is used by this lambda function to pass arguments in the context of tkinter:

```
lt = Label(c3,text='Controller')
lt.pack(side='top',expand=1,fill='x')
l = Label(c3,text='',bg='red')
l.pack(side='top',expand=1,fill='x')
lt2 = Label(c3,text='Velocity')
lt2.pack(side='top',expand=1,fill='x')
l2 = Label(c3,text='',bg='yellow')
l2.pack(side='top',expand=1,fill='x')
lt3 = Label(c3,text='Desired Heading')
lt3.pack(side='top',expand=1,fill='x')
l3 = Label(c3,text='',bg='blue')
l3.pack(side='top',expand=1,fill='x')

b2 = Button(c3,text='load',command=lambda:loadme(l,l2,l3))
b2.pack(fill='x', padx=2, pady=2)
root.mainloop()
```

Lesson 111. Event Binding

To bind widget "pk", describe the event and list the function to call. If the function is a method in a class, define the arguments as (self, event).

```
pk.bind('<Return>', self.view)
def view(self, event):
```

111.1 Bind to Main Window

Try binding to your main variable:

```
from Tkinter import *

main = Tk()

def leftKey(event):
    print "Left key pressed"

def rightKey(event):
    print "Right key pressed"

frame = Frame(main, width=100, height=100)
main.bind('<Left>', leftKey)
main.bind('<Right>', rightKey)
frame.pack()
main.mainloop()
```

The widget needs focus. See **stackoverflow**. For example:

```
1  def hey(event):
2      print('hey, hey, hey ... ')
3
4  import tkinter as tk
5
6  root = tk.Tk()
7  tk.Label(root, text="Hoshi is at my window").grid()
8  tw = tk.Text(root)
9  tw.grid()
10 tw.bind('<Return>', hey)
11 tw.focus_set()
12 tw.bind('<Left>' , hey)
13 root.mainloop()
```

Another example, event binding is used to select a line of text in a text widget.

```
def hey(event):
    print('hey, hey, hey ... ')
def get_line(event):
    print('getting line')
    print(tw.get('sel.first linestart', 'sel.last lineend'))
    print(tw.get('current linestart', 'current lineend'))

import tkinter as tk

root = tk.Tk()
tk.Label(root, text="Hoshi is at my window").grid()
tw = tk.Text(root)
tw.grid()
tw.bind('<Return>', hey)
#KP_Down
tw.focus_set()
tw.bind('<Left>' , hey)
tw.bind('<Shift-Up>', get_line)
root.mainloop()
```

Lesson 112. Removing and Hiding Widgets

calling `widget.destroy()` removes the widget and all its children.

calling `widget.grid_forget()` removes the widget from view but does get rid of it.

Note! For removing/hiding images, see the tk image problem in Lesson 121.

Lesson 113. Images

For a gif:

```
myimage = tk.PhotoImage(file=myfile)
tk.Label(root, image=myimage)
```

For a non-gif: I have written a class to convert an image file to a tk image objects.
`/Users/donaldelger/SpyderProjects/CourseBuilderR1/image.py`

Lesson 114. Getting Widgets (Images) to Cycle

Need to take care of the tk Image Problems; see 121

Create a widget variable name that is global.

Use either the `widget.destroy()` or `widget.grid_forget()`
(both will work ...)

Lesson 115. Power User Methods

115.1 Making Widgets Stretchable; Controlling widget size

Use the `grid_columnconfigure` method on the widget

```
root = Tk()
root.grid_columnconfigure(0, weight=1)
root.grid_rowconfigure(0,weight=1)
```

4.3. Configuring column and row sizes

Unless you take certain measures, the width of a grid column inside a given widget will be equal to the width of its widest cell, and the height of a grid row will be the height of its tallest cell. The `sticky` attribute on a widget controls only where it will be placed if it doesn't completely fill the cell.

If you want to override this automatic sizing of columns and rows, use these methods on the *parent* widget `w` that contains the grid layout:

`w.columnconfigure(N, option=value, ...)`

> In the grid layout inside widget `w`, configure column `N` so that the given `option` has the given `value`. For options, see the table below.

`w.rowconfigure(N, option=value, ...)`

> In the grid layout inside widget `w`, configure row `N` so that the given `option` has the given `value`. For options, see the table below.

Here are the options used for configuring column and row sizes.

Table 2. Column and row configuration options for the `.grid()` geometry manager

`minsize`	The column or row's minimum size in pixels. If there is nothing in the given column or row, it will not appear, even if you use this option.
`pad`	A number of pixels that will be added to the given column or row, over and above the largest cell in the column or row.

`weight`	To make a column or row stretchable, use this option and supply a value that gives the relative weight of this column or row when distributing the extra space. For example, if a widget `w` contains a grid layout, these lines will distribute three-fourths of the extra space to the first column and one-fourth to the second column:
	`w.columnconfigure(0, weight=3)`
	`w.columnconfigure(1, weight=1)`
	If this option is not used, the column or row will not stretch.

115.2 Finding/Changing the Attributes of Widget

This was the key for getting images to cycle:

Use `w.configure` method

```
def showImage(i):
    if i[0] > 2:
        i[0] = 0
    print(i)
    my_list = [image_tk, image_tk1, image_tkc]
    lbl1.configure(image=my_list[i[0]])
    i[0] = i[0] + 1
```

w.configure(option=value, ...)

Set the values of one or more options. For the options whose names are Python reserved words (`class`, `from`, `in`), use a trailing underbar: `'class_'`, `'from_'`, `'in_'`.

You can also set the value of an option for widget *w* with the statement

```
w[option] = value
```

If you call the `.config()` method on a widget with no arguments, you'll get a dictionary of all the widget's current options. The keys are the option names (including aliases like `bd` for `borderwidth`). The value for each key is:

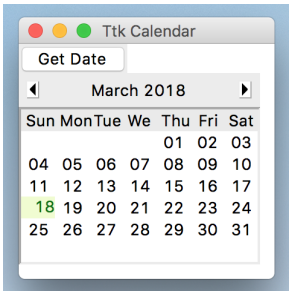
- for most entries, a five-tuple: (option name, option database key, option database class, default value, current value); or,
- for alias names (like `'fg'`), a two-tuple: (alias name, equivalent standard name).

The following message displays for 2 seconds.

115.3 Displaying a Message for 5 seconds (or 2)

```
top = tk.Toplevel(self, bg='powder blue')
font1 = "Helvetica 80 bold italic"
msg = 'Updated: pk = {}'.format(dct['pk'])

label1 = tk.Label(top, text=msg, width=len(msg), font=font1)
label1.grid(padx=30, pady=30)
top.after(2000, top.destroy)
```

Lesson 116. Calendar



This is source code for **ttkcalendar.py** which I modified to run in current version of tk.

Ref

Needed to revise the ttkcalendar.py module

Files are stored in learn/ttkcalendar folder.

```
import tkinter as tk
import ttkcalendar
import tkinter.simpledialog as sd
from tkinter import ttk

def don():
    def get_date():
        x = ttkcal.selection
        print('The selected date is: ', x)

    root = tk.Tk()
    root.title('Ttk Calendar')
    ttk.Button(root, text='Get Date', command=get_date).grid(sticky=tk.W)
    # firstweekday: 0 = monday, 1 = tuesday; 6 = sunday
    ttkcal = ttkcalendar.Calendar(firstweekday=6)
    ttkcal.grid()

    root.mainloop()

if __name__ == "__main__":
    don()
```

Lesson 117. Colors

Part XII: GUI CookBook

Lesson 118. Message that Self Destructs about 2 Seconds

```
import tkinter as tk
root = tk.Tk()
prompt = 'Timer is Done'
font1 = "Helvetica 80 bold italic"
label1 = tk.Label(root, text=prompt, width=len(prompt), font=font1)
label1.grid()

def close_after_2s():
    root.destroy()

root.after(2000, close_after_2s)
root.mainloop()
```

Written as a class, this algorithm looks like this:

```
class MyMessage(tk.Toplevel):
    """ Show a message that self destructs """
    def __init__(self, win, message, time):
        """ Show a message that self destructs

            Parameters:
                win (tkinter window): the window associated with top level
                message (string): the message
                time (integer): time in seconds for the message to display
        """
        super().__init__(win, bg='powder blue')
        self.grid()
        prompt = message
        font1 = "Helvetica 80 bold italic"
        label1 = tk.Label(self, text=prompt, width=len(prompt), font=font1)
        label1.grid(padx=30, pady=30)
        self.after(time*1000, self.close_win)

    def close_win(self):
        self.destroy()
```

Part XIII: Images in Python

Lesson 119. Summary

The 4 essential steps are

1. Import the python image library classes (line 2)
2. Open an image (line 8)
3. Resize the image as needed (line 10); this can also be done with the `.thumbnail` method.
4. Convert the image so that python can use it (line 11)
5. Reference the image anywhere that tkinter expects an image; note that the image must be saved as an attribute of the tk object (line 13)

```
1  import tkinter as tk
2  from PIL import ImageTk, Image
3
4  def talk():
5      print('hi there')
6
7  root = tk.Tk()
8  image = Image.open("temp.png")
9  new_size = int(22 / image.size[1] * image.size[0]), 22  # 22 pixels high
10 image2 = image.resize(new_size, Image.ANTIALIAS)
11 image3= ImageTk.PhotoImage(image2)
12 b = tk.Button(root, text="hello", image=image3, command=talk)
13 b.image = image
14 b.grid(padx=15, pady=15)
15 tk.Button(root, text='Text Button').grid()
16 root.mainloop()
```

Lesson 120. About Images in Python

There are three ways to use images in python

1. Display two color images (.xbm format) using the BitmapImage class.
2. Display full color images (.gif, .pgm, or .ppm) using the PhotoImage class.
3. Using image processing software (ImageMagick or PIL) to convert your image and then use options 1 or 2.

This **article** compares .gif to .png and .png.

.pgm: The acronym pgm stands for portable greymap". A .pgm file has a text-based image format for greyscale images.

.ppm: The PPM (portable pix map) image format is encoded in text that is human readable. See **this article**.

Lesson 121. Solving the tk Image Problem

The problem is that a tk image, when used in a function/class/etc. will not appear due to a bug in tk. The solution is to save the image as an attribute called *image* as follows:

```
my_widget.image = image_object
```

See this **effbot article** for more information.

Lesson 122. About PIL

The current version is called pillow; a fork of pil.

PIL allows you to process images with python.

This **webpage** presents a tutorial.

```
from PIL import Image
myimage = Image.open(filename)
myimage.load()

myimage.format  # file type
myimage.size    # pixel size
myimage.mode    # color model
```

```
from PIL import Image, ImageFilter
im = Image.open("lena.png")
im.thumbnail(size)
im.save(fname)
im.show()

im2 = im.filter(ImageFilter.CONTOUR)
im2.save('lena_contour' + '.jgp')
im2.show()
```

122.1 Filters

Original image

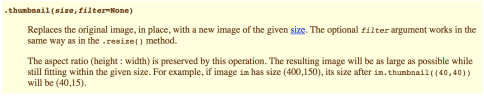
The contour filter

The emboss filter

Table 122.1: Image enhancement filters

Filter	Explanation
BLUR	
CONTOUR	
DETAIL	
EDGE_ENHANCE	
EDGE_ENHANCE_MORE	
EMBOSS	
FIND_EDGES	
SMOOTH	
SMOOTH_MORE	
SHARPEN	

122.2 Thumbnails (making images of given size)



`.thumbnail(size,filter=None)`

Replaces the original image, in place, with a new image of the given size. The optional `filter` argument works in the same way as in the `.resize()` method.

The aspect ratio (height : width) is preserved by this operation. The resulting image will be as large as possible while still fitting within the given size. For example, if image `im` has size (400,150), its size after `im.thumbnail((40,40))` will be (40,15).

Lesson 123. Example: Resizing (upwards)

The problem is that the image is not filling the window. The solution is to resize the image.

```
1 from PIL import Image
2 import numpy as np
3
4 mp = Image.open('mypicture.jpg')
5 mp.show()
6
7 s = np.array(mp.size)
8 size = s * 500/min(mp.size)
9 size = tuple(size.astype(int))
10
11 m2.show(mp.resize(size))
```

In line 7, the size method returns a tuple with the current image size.

Lines 7 to 9 use numpy's vector operations to convert the image size to have a minimum dimension of 500. Line 9 converts the numpy array to integer and then to a tuple.

Line 11 resizes the image and then displays it.

Lesson 124. ImageMagick

imagemagick.org

December 28, 2018. I reinstall using
`brew install imagemagick`

I had to screw around with brew to get it to link. The command that worked was
`brew link --overwrite imagemagick`

Then, I tested using a file on desktop
`magick 1.jpg 1.png`

This converted `1.jpg` to a `.png` file.

To resize into a 500 x 500 box while maintaining the aspect ratio, I used
`convert 1.jpg  -resize 500x500  1r.gif`

I had a lot of trouble with pdf conversions due to a black background. I had to add the `- flatten` option. The command I used was
`magick convert -density 300 -quality 100 -flatten 1fb.pdf 1fb.jpg`

Use the convert method; **link**

Update: 2017-11-08: Need to update to version 7 and use magick or magick-script.

Update: add the `-append` parameter to make multiple pages of images stack onto one page.

```
1 a = subprocess.run(['convert', '-density', '500', '-append', pdf_in, jpg_out])
```

The ImageMagick command-line tools exit with a status of 0 if no problems occur. If problems do occur, the exit status is 1.

124.1 What Works

```
sp = subprocess.run(['convert', '-density', '300', 'dog.pdf', 'dog.jpg'])
# sp.returncode = 0 for success in file conversion
```

The terminal command follows.

```
convert -density 5000 stand.pdf output.png
```

The density setting controls the image quality.

The python command follows.

```
subprocess.run(['convert', '-density', '4000',
                'stand.pdf','1.png'])
```

124.2 Quality from a pdf

June 20, 2019: The following worked well:
`convert -density 300 1.pdf -quality 100 -flatten 1.png`

Notes: need to trial and error convert; man helps some; -flatten is needed or background will be black–go figure?

So far, here are the rules I've developed. Convert the pdf to a jpg (not png). Set `-quality 90`. Try supersampling. A density of about 200 seems to work fine.

Lesson 125. MWE

see /Users/donaldelger/SpyderProjects/CourseBuilderR1/image.py

Part XIV: LaTeX Scripting

Lesson 126. Overview

I write a LaTeX file and use this syntax `[[topic]]` to indicate where I want to insert text into the file.

Next, I run a python script that executes the following steps.

1. Read the LaTeX file into a string variable *sv*
2. Replace the curly brackets and the [[topic]] using the string replace method four times. Some examples are

```
sv.replace('{', '{{')
sv.replace('[[', '{')
```

3. In python, build a dictionary *dict* that holds the keywords and replacement strings.
4. Run the string format method and pass this to the dictionary as follows.
   `sv.format(**dict)`
5. Write the string to a new file.

Part XV: Packages

Lesson 127. pint

units and dimensions

To install, I used `conda install -c conda-forge pint`

An example

```
import pint
ur = pint.UnitRegistry()
uf = '/Users/donaldelger/SpyderProjects/CourseBuilderR1/dbases/units_for_pint.txt'
ur.load_definitions(uf)

energy_per_volume = 32*ur.MJ/ur.L
area = 5.4*ur.m**2
area.ito_base_units()
print('area: ', area)
coefficient_drag = 0.32
density = 1.2*ur.kg/ur.m**3
speed = 30 * ur.m / ur.s
engine_efficiency = 0.4
```

Lesson 128. CoolProp

Fluid properties

To install, I used `pip install CoolProp`. The conda install did not work.

An example is:

```
from CoolProp.CoolProp import PropsSI

# Ask CoolProp water's heat capacity at 275.15 K (2C)
# and common Earth pressure 101325 Pa
heat_capacity = PropsSI('C','T',275.15,'P',101325,'INCOMP::Water')
#return 4182.587592215201
print(heat_capacity)

#Ask CoolProp water's mass density and viscosity with same conditions
density = PropsSI('D','T',275.15,'P',101325,'INCOMP::Water')
#return 1003.076063639064
print(density)

viscosity = PropsSI('V','T',275.15,'P',101325,'INCOMP::Water')
#return 0.0016507819947969723
print(viscosity)

kviscosity = viscosity/density
print(kviscosity)
```

To find saturation temperature or pressure, use a quality of 1.0 and one more property.

Lesson 129. Clipboard

Ref.

```
import clipboard
clipboard.copy("abc")  # now the clipboard content will be string "abc"
text = clipboard.paste()  # text will have the content of clipboard
```

Lesson 130. Calendar

```
import calendar
yy = 2018; mm = 9
print(calendar.month(yy, mm))  # display month
print(calendar.calendar(2018)) # display year
```

Note: in terminal, the command `cal` displays a calendar.

Part XVI: How to Program

Lesson 131. Nomenclature

An attribute is the value of a property; a “property of a property.”

Lesson 132. Procedural; Functional ;OOP

Procedural versus object oriented.

A component provides services to client (users) through its interfaces.

Zelle John Zelle. *Python Programming: An Introduction to Computer Science*. 2nd Edition. Franklin, Beedle & Associates Inc., 2010 asserts that the method of solving a hard problem is to break the problem down into a set of cooperating classes. The reason is that this approach reduces the complexity; allows the coder to focus on one thing at a time.

What does the class need to know?
What does the class need to be able to do?

The **attributes** of an object are the methods and parameters (instance variables).

Lesson 133. Testing

p. 217. Eric Matthes: Python Crash Course

Testing proves that your code works

A unit test verifies that one specific aspect of function's behavior works.

A test case is collection of unit tests that span all the possibilities.

Lesson 134. Version Control and GIT

This **book on git** explains everything very well. My top learning resource. I downloaded this book as a pdf file.

This article explains the main concepts of version control.

This **tutorial** was easy for me as a beginner.

p. 505, Eric Matthes: Python Crash Course
p. 471, Eric Matthes: Python Crash Course
p. 213, Cory Althoff: The Self Taught Programmer

Version control software give you the chance to store a copy of your program when your program is in a working state.

A version of your program that is saved is called a **commit**.

134.1 What, Why, Nomenclature

Version control is the management of information stored on a computer–e.g., documents, programs, web sites, and so on—in which changed files are identified and stored.

There are several common VCS, the two most common are

- SVN or Subversion which is described **here**
- GIT

Benefits of VC

1. Backup.
2. Synchronization.
3. Undo: both short-term and long term
4. Track Changes
5. Track Ownership
6. Try Something Out; both short term and long term (branch)

The nomenclature:

Basic Setup

Repository (repo): The database storing the files. Server: The computer storing the repo. Client: The computer connecting to the repo. Working Set/Working Copy: Your local directory of files, where you make changes. Trunk/Main: The primary location for code in the repo. Think of code as a family tree — the trunk is the main line.

Basic Actions

Add: Put a file into the repo for the first time, i.e. begin tracking it with Version Control. Revision: What version a file is on (v1, v2, v3, etc.). Head: The latest revision in the repo. Check out: Download a file from the repo. Check in: Upload a file to the repository (if it has changed). The file gets a new revision number, and people can "check out" the latest one. Checkin Message: A short message describing what was changed. Changelog/History: A list of changes made to a file since it was created. Update/Sync: Synchronize your files with the latest from the repository. This lets you grab the latest revisions of all files. Revert: Throw away your local changes and reload the latest version from the repository. Advanced Actions

Branch: Create a separate copy of a file/folder for private use (bug fixing, testing, etc). Branch is both a verb ("branch the code") and a noun ("Which branch is it in?"). Diff/Change/Delta: Finding the differences between two files. Useful for seeing what changed between revisions. Merge (or patch): Apply the changes from one file to another, to bring it up-to-date. For example, you can merge features from one branch into another. (At Microsoft this was called Reverse Integrate and Forward Integrate) Conflict: When pending changes to a file contradict each other (both changes cannot be applied). Resolve: Fixing the changes that contradict each other and checking in the correct version. Locking: Taking control of a file so nobody else can edit it until you unlock it. Some version control systems use this to avoid conflicts. Breaking the lock: Forcibly unlocking a file so you can edit it. It may be needed if someone locks a file and goes on vacation (or "calls in sick" the day Halo 3 comes out). Check out for edit: Checking out an "editable" version of a file. Some VCSes have editable files by default, others require an explicit command. And a typical scenario goes like this:

Alice adds a file (list.txt) to the repository. She checks it out, makes a change (puts "milk" on the list), and checks it back in with a checkin message ("Added required item."). The next morning, Bob updates his local working set and sees the latest revision of list.txt, which contains "milk". He can browse the changelog or diff to see that Alice put "milk" the day before.

134.2 How To

First, install git and create a GitHub account; see web for details. Once the software is installed

Step 1: Make a root folder XYZ. Add one or more files.

Step 2: Create Repo. In terminal, navigate to root to the root folder and initialize a git repo by typing
`git init`

Step 3: Stage. In terminal type, `git add <filename>`.

Step 4: ???Make a .gitignore file in folder XYZ. The contexts of the file are: `__pycache__/`

Lesson 135. Nomenclature for Paths and Files

pathname = mac name for the full path

I think your search for a "standard" naming convention will be in vain. Here are my proposals, based on existing, well-known programs:

A) C:\users\OddThinking\Documents\My Source\Widget**foo**.src

Vim calls it *file root* (:help filename-modifiers)

B) C:\users\OddThinking\Documents\My Source\Widget**foo.src**

file name or base name

C) C:\users\OddThinking\Documents\My Source\Widget\foo.**src** *(without dot)*

file/name extension

D) C:\users\OddThinking\Documents\My Source\Widget\foo.**src** *(with dot)*

also *file extension*. Simply store without the dot, if there is no dot on a file, it has no extension

E) **C:\users\OddThinking\Documents\My Source**\Widget\foo.src

top of the tree
No convention, git calls it *base directory*

F) C:\users\OddThinking\Documents\My Source**Widget\foo.src**

path from top of the tree to the leaf
relative path

G) C:\users\OddThinking\Documents\My Source**Widget**\foo.src

one node of the tree
no convention, maybe a simple *directory*

H) **C:\users\OddThinking\Documents\My Source\Widget**\foo.src

dir name

I) **C:\users\OddThinking\Documents\My Source\Widget\foo.src**

full/absolute path

Lesson 136. Commands

`git config user.name`: Show the Git username

`git config --list`: List all the configuration information

Part XVII: Engineering Stuff

Lesson 137. Fluid Properties

CoolProp is a C++ library that finds fluids properties and more.

PropsSI is a the python module for running CoolProp

```
# Import the PropsSI function
In [1]: from CoolProp.CoolProp import PropsSI

# Saturation temperature of Water at 1 atm in K
In [2]: PropsSI('T','P',101325,'Q',0,'Water')
Out[2]: 373.1242958476844
```

Names of properties

mass density: D, DMASS, Dmass
pressure P
mass vapor quality: Q
viscosity: V, viscosity

Lesson 138. Solving One Equation

When an equation cannot be solved by algebra, there are a variety of techniques ...

```
from scipy.optimize import fsolve

def my_f(x):
    return x**3 - 27.01

print(fsolve(my_f, 2))
```

Lesson 139. Types of Paths

There are three kinds of paths

1. Absolute: perhaps the best
2. Root Relative; relative to the root folder of site or project
3. Document Relative: avoid; hard to manage; can break

139.1 Latex: Root and Absolute Combination

see **newcommand* versus newcommand**

Lesson 140. Solving a Set of Nonlinear Equations

The following example shows how to solve three equations.
Source: `/Users/donaldelger/Documents/*A_C/BookCourses/FluidMechanics/tasks/_jupyter/15.py`

```
from scipy.optimize import fsolve


def equations(p, *args):
    """ Find the force to hold a horizontal nozzle """
    V1, V2, Fx = p  # parameters to be solve for
    rho, A1, A2, p1 = args
    mdot = rho * A1 * V1
    eq1 = p1*A1 + Fx - mdot * (V2 - V1)
    eq2 = p1 + rho * V1**2/2 - rho*V2**2/2
    eq3 = A1*V1 - A2*V2
    return (eq1, eq2, eq3)


def print_vars (vars):
    """ print values variables created with pint;
        vars = a string with variables names separated by commas
        example vars = ('F, m, acceleration')
    """
    vars = [var.strip() for var in vars.split(',')]
    for var in vars:
        print('{}: {}'.format(var, eval(var)))


# Specify the input parameters
rho, A1, A2, p1 = (
        1000, 50e-4, 10e-4, 2.5*100e3)
inputs = rho, A1, A2, p1


# Specify initial guess values of V1 ,V2, p2, Fx, Fy
guess_values = (1, 10, 1000.)

# Solve for the unknown parameters
V1, V2, Fx = fsolve(equations, guess_values, args=inputs)

# Print the results
print('\nInput Parameters')
print_vars('rho, A1, A2, p1')
print('\nOutput Parameters')
print_vars('V1, V2, Fx')
```

Listing 140.1: Example: Solving Three Equations

To solve a set of nonlinear equation, use fsolve from scipy.optimize

To find the roots of a polynomial, the command **roots** is useful. To find a root of a set of non-linear equations, the command **optimize.fsolve** is needed. For example, the following example finds the roots of the single-variable transcendental equation

$$x + 2\cos(x) = 0,$$

and the set of non-linear equations

$$\begin{aligned} x_0 \cos(x_1) &= 4, \\ x_0 x_1 - x_1 &= 5. \end{aligned}$$

The results are x = -1.0299 and x_0 = 6.5041, x_1 = 0.9084.

```
>>> def func(x):
        return x + 2*cos(x)

>>> def func2(x):
        out = [x[0]*cos(x[1]) - 4]
        out.append(x[1]*x[0] - x[1] - 5)
        return out

>>> from optimize import fsolve
>>> x0 = fsolve(func, 0.3)
>>> print x0
-1.02986652932

>>> x02 = fsolve(func2, [1, 1])
>>> print x02
[ 6.5041  0.9084]
```

Ref: search for fsolve

Lesson 141. Random Numbers

Random Module

Numpy Random

```
import random
import numpy as np

random.sample(set('abcde')

myset = np.around(np.random.uniform(0, 2*377, 4), decimals=0)

ans = [8, 69, 43, 38, 23]
random.shuffle(ans)
```

Lesson 142. Permutations and Combinations

In English we use the word "combination" loosely, without thinking if the **order** of things is important. In other words:

"My fruit salad is a combination of apples, grapes and bananas" We don't care what order the fruits are in, they could also be "bananas, grapes and apples" or "grapes, apples and bananas", its the same fruit salad.

"The combination to the safe is 472". Now we **do** care about the order. "724" won't work, nor will "247". It has to be exactly **4-7-2**.

So, in Mathematics we use more *precise* language:

- When the order doesn't matter, it is a **Combination**.
- When the order **does** matter it is a **Permutation**.

So, we should really call this a "Permutation Lock"!

In other words:

A Permutation is an **ordered** Combination.

To help you to remember, think "**P**ermutation ... **P**osition"

213

The following code with Python 2.6 and above ONLY

First, import `itertools` :

```
import itertools
```

Permutation (order matters):

```
print list(itertools.permutations([1,2,3,4], 2))
[(1, 2), (1, 3), (1, 4),
(2, 1), (2, 3), (2, 4),
(3, 1), (3, 2), (3, 4),
(4, 1), (4, 2), (4, 3)]
```

Combination (order does NOT matter):

```
print list(itertools.combinations('123', 2))
[('1', '2'), ('1', '3'), ('2', '3')]
```

Bibliography

Zelle, John. *Python Programming: An Introduction to Computer Science*. 2nd Edition. Franklin, Beedle & Associates Inc., 2010.

Lesson 143. Python Modules and Packages

Some references are:

realpython.com

stackoverflow.com

tutorialspoint

another tutorial ...

Lesson 144. Exceptions

144.1 Raise an Exception

To raise an exception:

```
raise ValueError('A very specific bad thing happened.')
```

stackoverflow

Part XVIII: Useful Packages

Lesson 145. Roman Numeral Converter

help(roman)

pypi site

import roman

fromRoman(s) convert Roman numeral to integer

toRoman(n) convert integer to Roman numeral

Part XIX: Python Connections

Lesson 146. Python files

To find source code for python functions: **stack overflow**

To find the location of python modules: **stack overflow**

Lesson 147. Package: Build Your Own

147.1 Rationale for Using a Package

Using a package makes organization of files easier. Organization saves times, ...

programiz.com says the following:

> We don't usually store all of our files in our computer in the same location. We use a well-organized hierarchy of directories for easier access.
>
> Similar files are kept in the same directory, for example, we may keep all the songs in the "music" directory. Analogous to this, Python has packages for directories and modules for files.
>
> As our application program grows larger in size with a lot of modules, we place similar modules in one package and different modules in different packages. This makes a project (program) easy to manage and conceptually clear.
>
> Similar, as a directory can contain sub-directories and files, a Python package can have sub-packages and modules.
>
> A directory must contain a file named `__init__.py` in order for Python to consider it as a package. This file can be left empty but we generally place the initialization code for that package in this file.

147.2 Modular Programming

This quote is from **ref**:

> Modular programming refers to the process of breaking a large, unwieldy programming task into separate, smaller, more manageable subtasks or modules. Individual modules can then be cobbled together like building blocks to create a larger application.
>
> There are several advantages to modularizing code in a large application:
>
> Simplicity: Rather than focusing on the entire problem at hand, a module typically focuses on one relatively small portion of the problem. If you're working

on a single module, you'll have a smaller problem domain to wrap your head around. This makes development easier and less error-prone.

Maintainability: Modules are typically designed so that they enforce logical boundaries between different problem domains. If modules are written in a way that minimizes interdependency, there is decreased likelihood that modifications to a single module will have an impact on other parts of the program. (You may even be able to make changes to a module without having any knowledge of the application outside that module.) This makes it more viable for a team of many programmers to work collaboratively on a large application.

Reusability: Functionality defined in a single module can be easily reused (through an appropriately defined interface) by other parts of the application. This eliminates the need to recreate duplicate code.

Scoping: Modules typically define a separate namespace, which helps avoid collisions between identifiers in different areas of a program. (One of the tenets in the Zen of Python is Namespaces are one honking great idea—let's do more of those!)

Functions, modules and packages are all constructs in Python that promote code modularization.

147.3 Package

Any Python file is a module, its name being the file's base name without the .py extension. A package is a collection of Python modules: while a module is a single Python file, a package is a directory of Python modules containing an additional `__init__.py` file, to distinguish a package from a directory that just happens to contain a bunch of Python scripts. Packages can be nested to any depth, provided that the corresponding directories contain their own `__init__.py` file.

ref: module versus package

147.4 How to Set up a Package

Minimalist Tutorial

Hitchhikers Guide

uoftcoders

See **ref**.

Lesson 148. Namespaces and scope

Here **programiz.com** explains a namespace.

A name, aka an identifier, is a name for an object.

A namespace is a set or collection of names.

Lesson 149. Logging

ref

https://realpython.com/python-logging/

a. Get the Name of the module

```
logger= logging.getLogger('HoshiLogger')
logging.basicConfig(level=logging.INFO)
logger.info('you have reached line 38')
```

b. Format the Log Message

Method 1: Use the basicConfig(**kwargs) method to configure the format of the logging:

```
logging.basicConfig(format='%(asctime)s - %(message)s'
                    , datefmt='%d-%b-%y %H:%M:%S')
```

This logging message:

INFO (02/03/19, 06:19, line 38, log1): You need to take a break from coding, dude.

was printed out which this code:

```
logger = logging.getLogger(__name__)
logging.basicConfig(format='%(levelname)s (%(asctime)s, line %(lineno)d,
%(module)s): %(message)s', level=logging.INFO, datefmt='%m/%d/%y, %I:%M')
logger.info('You need to take a break from coding, dude.')
```

c. Set up a Module for Logging

Fig. 149.1 shows how to set up a module for logging.

for the application to route their log messages. For this reason, it is a convention for each module to simply use a logger named like the module itself. This makes it easy for the application to route different modules differently, while also keeping log code in the module simple. The module just needs two lines to set up logging, and then use the named logger:

```
import logging

log = logging.getLogger(__name__)

def do_something():
    log.debug("Doing something!")
```

That is all there is to it. In Python, `__name__` contains the full name of the current module, so this will simply work in any module.

Figure 149.1: Setting up a module for logging

d. Using Logging in Multiple Modules

ref

The recommended way to setup logging is to not define any handlers nor logging levels into the modules, but define all the configuration—i.e., the handlers—in the main file **ref**.

Bibliography

Zelle, John. *Python Programming: An Introduction to Computer Science*. 2nd Edition. Franklin, Beedle & Associates Inc., 2010.

Index