Learning Python – Chris Morgan (11/17/2017)

To install Python 2 plus various useful modules and examples go to: www.helloworldbook2.com
To install just Python 2.7.x or Python 3.x go to: https://www.python.org/downloads/

IDLE (Interactive Development and Learning Environment) -- This program opens an interactive window for you to enter Python commands and create programs. Use `exit()` or `quit()` built-in function to quit IDLE.

Interactive command window (IDLE)

```
>>>print("Hello World" *5)
>>>print(25*5)
>>>print(12345 * 6789)
>>>print("cat" + " " + "dog")
>>>print(25**2)
>>>print(3/2)
>>>print(3.0/2)
>>>x = 5
>>>print(x * 5)
>>>y = 3
>>>print(x + y)
>>>print((x + y)**2)
```

Creating your first program – Program1.py

Step 1. Open the program editor screen in IDLE: File ➔ New Window (or press <Ctrl+N>)

Step 2. Enter the following lines of Python code in the editor window:

```
print("Hey, Luke! " *20)
x = 5
print(x)
x = x + 5
print(x)
print("End of Program")
```

Step 3. Save the program to the hard disk
File ➔ Save As ➔ Program1.py (or, press <Ctrl+S>)
Step 4. Run the program
Run ➔ Run Module (or, press <F5>)

Python help online

```
http://docs.python.org/2/
http://docs.python.org/2/library
```

PyCharm Development Environment

PyCharm is a more sophisticated development environment to learn and use Python. The EDU edition is based on the free 'Community Edition', but comes with additional learning material to help the beginner.

```
http://www.jetbrains.com/pycharm-edu/concepts
```

Create your second program – Program2.py (GuessMyNumber)

Step 1. Open the program editor screen in IDLE: File ➔ New Window (or press <Ctrl+N>)

Step 2. Enter the following lines of Python code in the editor window:

```
import random
secret = random.randint(1,20)
guess = 0
tries = 0
print("HEY! I'm the Dread Pirate Luke, and I have a secret!")
print("It is a number from 1 to 20. I'll give you 6 tries to guess it.")
while guess != secret and tries < 6:
    guess = int(input("Make a guess: "))
    if guess < secret:
        print("Too low, landlubber!")
    elif guess > secret:
        print("Too high, you scurvy dog!")
    tries = tries + 1
if guess == secret:
    print("Avast! Ye got it! Found my secret, ye did.")
else:
    print("No more guessses!)
print("The secret number was ", secret)
```

Step 3. Save the program to the hard disk

File ➔ Save As ➔ Program2.py (or, press <Ctrl+S>)

Step 4. Run the program

Run ➔ Run Module (or, press <F5>)

A word about Modules (ways to import a module and it's functions/classes)

`import module` (where 'module' is something like '`math`' or '`webbrowser`' or '`pygame`')

or

`import module1, module2, module3` (import multiple modules)

or

`from module import function` (import just one function)

or

`from module import func1, func2, func3` (import several functions)

or

`from module import *` (import all functions & classes from module)

```
import webbrowser as wb                    (change namespace, for coding convenience)
import datetime as dt
```

Example 1

```
import math
print(math.pi)
print(math.e)
print(math.sqrt(25))
```

Or, just import the `sqrt` *function. Then, you don't have to refer to the module name to use it.*

```
from math import sqrt
print(sqrt(25))
```

Example 2

```
import webbrowser
webbrowser.open("http://www.thegreatcourses.com")
```

Example 3

```
import  time, calendar,sys
print(time.ctime())
print(calendar.month(2016,11)
calendar.prcal(2017)
sys.path
sys.path.append("new directory")
```

A word about packages (bundles of modules) and downloading other modules.

- There is a file in every package directory with the special name (the contents of the file is unimportant. It can be a blank (empty) file): `__init__.py`
- The 'Python Package Index' (PyPI) has the official index of all Python packages. It is located here:
  http://pypi.python.org/pypi
- Google "PyPi Ranking" to see a list of popular 3rd party packages
- Installing a new module from the CMD command line (not in IDLE). Use the following two commands:
  `python get-pip.py` (this installs the installer module 'pip'. It is a one-time thing.)
  `python –m pip install pkgname` (then, this instruction uses 'pip' to install the module you want)

How to find out about installed modules

```
help('modules')     # to see a list of all the installed modules on your computer
import math         # activate (import) the math module
help(math)          # show a list of all the functions in 'math'
help(math.trunc)    # get info on the 'trunc' function in 'math'
dir(math)           # show list of the functions in 'math'
sys.builtin_module_names   # show a list of built-in modules (not .py files)
```

Create your third program – Program3 (NameChanger)

This program will take all the JPG files in the directory you specify and rename them.

Step 1. Open the program editor screen in IDLE: File ➔ New Window (or <Ctrl+N>)

Step 2. Enter the following lines of Python code in the editor window:

```
import os
dirname = input("Enter a directory: ")
os.chdir(dirname)
dirlist = os.listdir()
picture_number = 0

for f in dirlist:
```

```
    if f.upper().endswith("JPG") and not f.startswith("Summer"):
      picture_number = picture_number + 1
      newname = ("SummerVacation_" + ("0000" + str(picture_number))[-3:] + ".jpg")
      os.rename(f, newname)
      print("Picture " + str(picture_number) + " renamed")
  dirlist = os.listdir()
  for f in dirlist: print(f)
  print(str(picture_number) + " pictures renamed")
```

Step 3. Save the program to the hard disk

File ➔ Save As ➔ Program3.py (or, <Ctrl+S>)

Step 4. Run the program

Run ➔ Run Module (or, <F5>)

Here are some more short programs you can write to learn about additional modules:

Program 4

```
from turtle import forward, left     # other functions: backward, right, penup, pendown
for i in range(100):
  forward(3*i)
  left(90)
input()      #wait to clear the screen until user presses <Enter> key
```

Program 5

```
from turtle import forward, left, right
def drawsquare(size = 100):
  forward(size)
  left(90)
  forward(size)
  left(90)
  forward(size)
  left(90)
  forward(size)
  left(90)
for i in range(12):
  drawsquare(300)
  right(30)
input()
```

Program 6

```
from urllib.request import urlopen
file = urlopen("http://helloworldbook2.com/data/message.txt")
message = file.read()
print(message)
```

Program 7

```
import easygui
easygui.msgbox("Hello there!")
```

Program 8

```
import easygui
```

```
response = easygui.msgbox("Hello there!")
flavor = easygui.buttonbox("Choose a flavor:", choices = ["Vanilla","Chocolate","Strawberry"])
easygui.msgbox("You picked " + flavor)
```

You can also try other functions instead of easygui.buttonbox: **easygui.enterbox easygui.choicebox**

Various Modules (libraries of functions)

Here is a list of some common modules you can explore in Python:

math	time	turtle	os	statistics	calendar	ipython
matplotlib	scipy	numpy	pygame	random	shutil	glob
pickle	urllib	urllib2	tkinter	serial	datetime	sys
ssl	webbrowser	graphics	pygal	easygui	pyqt4	cmd

Reserved words

True	class	(exec)	in	(print)
False	continue	finally	is	raise
None	def	for	lambda	return
And	del	from	nonlocal	try
As	elif	global	not	while
Assert	else	if	or	with
Break	except	import	pass	yield

Built-In Functions (There are 70. Below are just a select few)

dict()	id()	min()	round()	chr(i) char from int
dir()	input()	next()	slice()	ord(c) int from char
eval()	int()	object()	sorted()	
exec()	len()	open()	str()	
float()	list()	pow()	sum()	
format()	locals()	print()	tuple()	
help()	max()	range()	type()	

Math Operators: +, -, *, /, // (int div), %, **

Math Library

Pi	tan(x)	log10(x)
E	asin(x)	exp(x)
sqrt(x)	acos(x)	ceil(x)
sin(x)	atan(x)	floor(x)
cos(x)	log(x) e.g. ln(x)	

String Methods

s.capitalize()	s.endswith(str)	s.rjust(w)	s[m,n]
s.lower()	s.center(n)	s.ljust(w)	s.splitlines()
s.upper()	s.center(n,fill)	s.title()	s.isdigit()
s.find(str)	s.count(str)	s.split(‘,’)	s.isdecimal()
s.strip()	s.index(str)	s.join()	s.isalnum()
s.startwith(str)	s.replace(a,b)	s.zfill(w)	s.isalpha()
s.partition(’,’)	s.replace(a,b,m)		

Formatting output

```
print('My name is %s. I have %d books worth $%.2f.'%('Bill',4,4*9.95)
My name is Bill. I have 4 books worth $39.80.
print('Amount is: %10.2f per bag'%(62.12567))
Amount is:      62.12 per bag
print('Amount is: %-10.2f per bag'%(62.12567))
Amount is: 62.12      per bag
print('%-15s%-15s%5s'%('Name','Rank','S/N'))
print('%-15s%-15s%5s'%('----------','----------','-----'))
Name           Rank             S/N
----------     ----------     -----
s–string, c-char from an int, d-decimal, i-int, f-floating point dec, e-floating point exp
r'c:\file.txt' raw text, \t tab, \n newline, \f formfeed, \' or \" quote, \\ backslash
'''here is some text'''  multi-line block of text
```

Lists []

l = list() or l = []	create an empty list l
l=['one','two','three']	creates a list l with 3 items
l.append('four')	add the item 'four' to the list l
l.insert(i,x)	inserts x at index position i
l.pop(i)	removes the ith item from l. Also, returns the ith item
l.remove('one')	removes the first occurrence of 'one' from l
l.extend(l2)	appends list l2 to list l
l.count()	number of items in the list l
l.sort()	sort the items in list l
l.sort(key=func, reverse=True)	returns a list sorted in reverse order using the value returned by function func()
l.reverse()	reverses the sorting of the items in list l
l.clear()	deletes all the items in the list
l.copy()	generates a copy of the list
del l[i]	removes the ith item from l
del l[i,j]	removes the ith thru (j-1)th items from list l
len(l)	returns the number of items in list l
max(l)	returns the max value found in list l
min(l)	returns the min value found in list l
list(s)	returns s as a list of characters
s.split()	splits string s into a list of words
', '.join(l)	returns a comma delimited string from the list l
''.join(l)	joins all the items in l into a single string

Tuples ()

t = ()	create a tuple
t = ('apple','banana','pear')	creates a tuple with 3 items
t.count()	number of items in the tuple
t.index('pear')	index of the item in tuple t having the value 'pear'

Notes: 1. Tuples can't be changed: no editing, sorting, appending, or reversing of the entries (i.e., immutable)

2. Functions can return a tuple

Dictionaries {} associative array

d = dict() or d = {}	create an empty dictionary d
d = {'one':1,'two':2}	creates a dictionary with 2 entries. JSON format.
d['one'] = 'uno'	creates or updates an entry in the dictionary with the key 'one' and a value of 'uno'
a = d['one']	assign 'uno' to variable a
d.get('one','-')	returns the value associated with the key 'one'. If not found, return a dash ('-').
len(d)	number of items in the dictionary d
if 'one' in d	True if 'one' is a key in d
list(d.values())	creates a list of all the values found in d
list(d.keys())	creates a list of all the keys in d
d.items()	returns a list of tuples, where each tuple has two items: a key and its value
d.clear()	deletes all entries in d
del d['one']	deletes the entry having key 'one' in d

Conditionals

```
if a == b:
    print a
elif a > b:
    print b
else:
    print a
```

```
names = ['Tom','Dick','Harry']
if 'Tom' in names:
    'Hi Tom'
if 'Tom' not in names:
    'No Tom here'
```

Loops

`for x in [1,2,3,4,5]:` `  print(x)`	Loop through all the items in the list 1, 2, …, 5 and print the value at each iteration
`for x in range(10):` `  print(x)`	Loop 10 times and print out the iterator value 0, 1, 2, …, 9
`while a < b:` `  print(b-a)`	Loop any number of times while a is less than b. Whenever this becomes false, exit the loop
`for c in name:` `  print c`	prints a list of the characters in the string 'name'
`for name in names:` `  print(name)`	prints out all the names found in the list 'names'
`for k,v in d.items()`	loops thru the dictionary d and returns a tuple pair for each entry, assigning the key to k and the value to v
`continue`	When encountered, stop further execution of the current iteration and go to the next one
`break`	When encountered, stop the looping and go to the next instruction after the looping structure

Functions

- Keyword '*def*' starts a function definition; followed by the function name, a list of parameters (in parentheses), and a semicolon
- Parameters are optional
- Function may return a value, but it is optional
- Variables defined in a function are 'local' (i.e., not seen outside the scope of the function)

- You can reference a global variable inside a function, but you can't change its value unless you declare it as 'global' first. Otherwise, if you assigned a global variable a value in a function it would actually just create a local variable that has the same name as the global variable.
- You can return multiple values by putting them in parentheses, separated by commas. The function would then return a tuple, instead of just a value.

```
def funcname(parm1, parm2):
    a = parm1 * parm2
    return a                    # return a value
def funcname():
    return (1,2,3,4)            # returns a tuple, instead of a value
```

Errors (see: https://docs.python.org/3/library/exceptions.html)

```
try:
    code to execute that might produce an error
except errtype:
    error handler for error type errtype
except errtype1, errtype2:
    error handler for error types errtype1 or errtype2
except:
    error handler for any error type
else:
    code to execute if no errors produced in the try block
finally:
    code to execute regardless of an error occurring
```

Common error types: FileNotFoundError, ZeroDivisionError, ValueError, AssertionError, ConnectionError, TypeError

`raise` Forces an error to be raised

`assert` expr, msg Causes *AssertionError* to be raised with a message 'msg', if expr evaluates to False

Classes

```
class Car():
    '''A simple class to demonstrate how a class is defined.'''
    def __init__(self, parm1, parm2):
        '''Initialize attributes (properties)'''
        self.color = parm1
        self.year = parm2

    def func1(self):
        '''Comments on function 1'''
        print(self.color.title() + " is the value of parm1")

myCar = Car('Red',1960)         #creates an object based on the class Car, with two parms.
print("The color of myCar is " + myCar.color + " and the year built is " + myCar.year)
myCar.func1()                   #executes method func1() of myCar
```

- The special function `__init__()` is the constructor for the class. It is executed automatically when an instance of the class is created.
- Every function defined in the class must have the first parameter = 'self'.
- By convention, class names start with an initial cap and instance objects with lower case.
- To define a class that inherits from another class, include the super class name in the parentheses of the class name. Then, add a function call in the __init()__ method, `super().__init__(parm1, parm2)`, that references the parameters in the parent class.

- You can 'hide' internal functions in a class that are not accessible from outside the class definition by starting the name with a single underscore. They will not be included in an 'import'.

Files

`f = open(r'c:\python\file.txt','r')`	Open a file in 'read' mode
with open("myfile.txt") as f:	Alternative form of open. Automatically closes file after execution of the code block following this line.
open modes: r – read, w – write, a – append, r+ - read and write rb – read bytes, wb – write bytes	
`t = f.read()`	Reads the whole file into the variable t
`t = t.read(n)`	Reads n bytes to t
`t = f.readline()`	Only reads one line of the file
`t = f.readlines()`	Reads the whole file into a list of lines
`for line in t:` `  print line`	Iterates through the list of lines t and prints out each one.
`for line in f.readlines():` `  print line`	Same as above, but only reads the lines from the file as needed; skips reading the whole file at once.
`f.write(s)`	Writes string s to file f
`f.writelines(l)`	Writes a list l to file f
`f.close()`	Closes file f
`f.tell()`	Returns the current pointer position in the file f
`f.seek(m)`	Sets the pointer (cursor) to position m in file f
`f.seek(m,n)`	Sets the current file position to m. If n=0, from front of file. If n=1, from current position. If n=2, from end of file.
`f.truncate()`	Truncates the file to the current cursor position
`f.truncate(m)`	Truncates the file f to m bytes
`f.closed`	True or False (has file f been closed yet)
`f.name`	Name of the file that was opened as f
`f.mode`	Mode used to open the file

SQLite3 – Program9 (doesn't work on RPi)

```
import os
from sqlite3 import connect
with connect(r'temp.db') as conn:
    curs = conn.cursor()
    if not os.path.exists('temp.db'):
        curs.execute('create table emp (who, job, pay)')
        prefix = 'insert into emp values '
        curs.execute(prefix + "('Bob', 'dev', 100)")
        curs.execute(prefix + "('Sue', 'dev', 120)")
        curs.execute(prefix + "('Jim', 'tester', 80)")
        conn.commit()
    curs.execute("select * from emp where pay >= 100")
    print('%-10s%-10s%-10s'%('Name','Pos','  Rate'))
    print('%-10s%-10s%-10s'%('----','---','--------'))
    for (who, job, pay) in curs.fetchall():
        print('%-10s%-10s$%7.2f'%(who, job, pay))

    result = curs.execute("select who, pay from emp")
    rows = result.fetchall()
    for row in rows:
        print(row)
```

```
    query = "select who, job from emp where job = ?"
    curs.execute(query,('dev',))
    for row in curs.fetchall():
        print(row[0],"\t",row[1])
```

conn.close()	Closes the connection
conn.commit()	Commits any pending transaction
conn.rollback()	Rollback pending transaction
conn.cursor()	Returns a new cursor object
curs.description	Sequence of seven-item sequences
curs.rowcount	Specifies the number of rows
curs.callproc(procname, params)	Calls a stored procedure
curs.close()	Closes the cursor
curs.execute(operation, params)	
curs.executemany(operation, seq_of_params)	Prepare and execute a database operation
curs.fetchone()	Fetch the next row of a query result set
curs.fetchmany(n)	Fetch next n set or rows of a query, as tuples
curs.fetchall()	Fetch all rows of a query, as a list of tuples
Date(yr, mon, day)	Constructs an object holding a date value
Time(hr, min, sec)	Constructs an object holding a time value
None	Sql NULL

Count Letters in a Word - Program 10

Program to count the occurrence of each letter found in the word 'brontosaurus' and store this list of letters as a 'dictionary'. Then, sort it by frequency of occurrence.

```
def byFreq(pair):
    return pair[1]    #return the second item in the list 'pair'

d = dict()
word = 'brontosaurus'
for c in word:
    d[c] = d.get(c,0) + 1
l = list(d.keys())
l.sort()

print('----------------1------------------------')
for a in l:
    print(a, d[a])

print('----------------2------------------------')
for k,v in sorted(d.items()):
    print(k,v)

print('----------------3------------------------')
t = list(d.items())
t.sort(key=byFreq,reverse=True)     #sort by second item in the list, in reverse order
for i in range(len(t)):
    char, count = t[i]
    print(char, count)

print('----------------4------------------------')
tmp = list()
for k,v in d.items():
    tmp.append((v,k))
```

```
for k,v in sorted(tmp):
    print(v,k)
```

Count Words in a File - Program 11

Program that counts the lines and words in a text file.

```
w = list()
d = dict()
linecount = 0
wordcount = 0
unwantedChars = [",",";",".","?",":",'"',"!","(",")","[","]"]
unwantedChars = ' ,;.?:"!()[]'   #works the same as the prev line
f = open("MobyDick.txt",'r')
for line in f.readlines():
    words = line.split()
    for word in words:
        for char in unwantedChars: word = word.replace(char,"")
        if word.startswith("'") or word.endswith("'"):
            word.strip("'")
        if word.startswith("-"): word = word.replace("-","")
        word = word.lower()
        d[word] = d.get(word,0) + 1
        wordcount = wordcount + 1
    linecount = linecount + 1
    #if linecount >= 1000: break
f.close()

words = list(d.keys())
words.sort(key=str.lower)
for word in words:
    if d[word] > 100: print(d[word],word)
#    if len(word) > 15 and not "-" in word: print(d[word],word)
#    print('%4d  %s'%(d[word],word))
print()
print("Unique words: ",len(d))
print("Wordcount: ",wordcount)
print("Linecount: ",linecount)
```

Resources:

www.python.org/downloads	To download either Python 2.x or 3.x
https://docs.python.org/3/library/functions.html	Official Python documentation for built-in functions
Python for Everybody – Exploring Data in Python 3, Charles Severance	http://www.py4e.com
Python Programming: An Introduction to Computer Science 3rd Ed, John Zelle	http://fbeedle.com http://mcsp.wartburg.edu/zelle/python
Hello World! 2nd Ed, Warren and Carter Sande	https://www.manning.com/books/hello-world-second-edition
Python Pocket Reference 5th Ed, Mark Lutz	O'Reilly, Inc.
Python Crash Course, Eric Matthes	No Starch Press, Inc., www.nostarch.com
Learn Python the Hard Way, Zed A. Shaw	Addison-Wesley, https://learnpythonthehardway.org/
EasyGUI	https://media.readthedocs.org/pdf/easygui/master/easygui.pdf
Pygame Doc	http://www.pygame.org/docs/tut/PygameIntro.html

PEP stands for **Python** Enhancement Proposal. A **PEP** is a design document providing information to the **Python** community, or describing a new feature for **Python** or its processes or environment. The **PEP** should provide a concise technical specification of the feature and a rationale for the feature.

Other topics…

- python --version (at the Windows command prompt)
- __name__ and "__main__"
- compiling with pycompile.compile()
- JSON files
- graphics.py
- .py vs .pyw vs .pyc
- Tkinter
- Pygame