Python Learning

The following exercises are based on the book “Beginning Python: From Novice to Professional”.

Hopefully you will set your own schedule and find someone to whom you can be held accountable for the work and demo what you have done. If you have suggestions please let me know.

Chapter 1: Instant Hacking: The Basics

Topics:

- Variables
- Strings
- Operators
- Functions
  - print()
  - input()
  - raw_input(), input() (Python 3.7)
- Modules (Import)
  - math
    - sqrt()
    - floor()
- Comments

Tasks:

- ❏ Complete Reading from Beginning Python Chapter 1
- ❏ Install PyCharm
- ❏ Create a new Python project
- ❏ Create a variable
- ❏ Ask a user for their name store this in the variable you created. Hint: input()
- ❏ Print out the name you were given. Hint: print()
- ❏ Ask a user for two numbers, store the numbers in two variables.
  - ❏ Add the two numbers together and print out a message: “Your numbers added together are: #”
  - ❏ Multiply the two numbers and print out a message: “Your numbers multiplied are: #”
- ❏ Ask a user for their first name. Then ask the user for their last name. Store these names in two variables called firstName and lastName. Put the names together and print them out on the screen. Hint: +
- ❏ Use the previous variables firstName and lastName and print out the names on two separate lines. Hint: /n

❏ Use the previous variables and print out the last letter of the firstName and lastName.
❏ Use the previous variables and print out the first letter of the firstName and lastName.
❏ Add comments to your program so someone else will know how it works.
❏ Demo your answers in the Python class

Demo:

What did you do? Show the completed program.
How did you do it? Walk through the code.

Chapter 2: Lists and Tuples

Topics:

Lists
Tuples

Tasks:

- ❏ Complete Reading from Beginning Python Chapter 2
- ❏ Create a new Python Project (or file) for this week
- ❏ Create a list of Presidents of the United States called presidents
  - ❏ Print out the list
  - ❏ Print the first President in the list
  - ❏ Print the last President in the list
  - ❏ Print the 5th President in the list
  - ❏ Print the list from the 3rd President in the list to the last
- ❏ Reverse the order of the list and name the new list reversedpresidents
  - ❏ Print both lists from the 6th to the 9th Presidents in the lists
- ❏ Make a list called fibonacci of the first 10 numbers in the Fibonacci series
- ❏ Create a list called numbers of numbers 1-10
  - ❏ Add the two lists together creating a new list called together
  - ❏ Check to see if 21 is in the list together
  - ❏ Check to see if 300 is in the list together
  - ❏ Print the length of the list together
  - ❏ Print the max and min numbers in the list together
- ❏ Create a tuple called fruits with 4 fruits in the tuple
  - ❏ Print the tuple
  - ❏ Print the 3rd fruit from the tuple

Quiz:

What is the difference between a list and a tuple?
What is the syntax to create a list?
What is the syntax to create a tuple?
How do you retrieve the last element of a list if you don't know how many elements are in the list?

Demos:

What did you do? Show the completed program.
How did you do it? Walk through the code.

Chapter 3: Working with Strings

Topics:

Strings with variables
String slicing
String formatting

Tasks:

- ❏ Complete Reading from Beginning Python Chapter 3
- ❏ Create a string variable that takes values for a name and birthday and prints it out.
    - ❏ For example: “[name] Birthday is on [birthday]” See “String Formatting: Short Version”
- ❏ Create a string with the first paragraph from the Declaration of Independence.
    - ❏ Print only the last 5 words from the above paragraph.
    - ❏ Count how many words are in the paragraph and print that out.
- ❏ Create a table (see listing 3.1 in the book) that displays 5 QA Engineers and their squad names. Engineer name should be left aligned and squad is right aligned. Include a header with column names.
- ❏ Create a list called list1 with values: [97, 24, 50, 56, 3, 82, 14, 38, 41, 65]
- ❏ Create a tuple called tuple1 with values: (77, 66, 99, 11, 22, 88, 44, 33, 55)
    - ❏ Print list1 sorted in descending order
    - ❏ Get the 3 largest numbers in list1 and print them out
    - ❏ Add the values from tuple1 to list1. Get the smallest value from the new list and print it out.
    - ❏ Add the values from tuple1 to list1. Get the second largest value from the new list and print it out.
    - ❏ Add tuple1 to list1 without modifying either list. Get the smallest value and print it out.

Demos:

What did you do? Show the completed program.
How did you do it? Walk through the code.

Chapter 4: Dictionaries: When Indices won't do

Topics:

Mappings
String formatting with Dictionaries
Dictionary methods

Tasks:

- ❏ Create an empty dictionary.
  - ❏ Print out the empty dictionary.
- ❏ Create a dictionary of Domo Holidays. https://domo.domo.com/kpis/details/1024668732
  - ❏ Retrieve and print Thanksgiving and Memorial Day's dates.
  - ❏ Update the Domo Holidays so we get the entire week of Thanksgiving off.
  - ❏ Print out the new Domo Holidays.
  - ❏ Add a new Holiday of your choosing.
  - ❏ Print out the new Domo Holidays.
- ❏ Create a dictionary of QA Engineers and their squads. (pick 5-10): https://domo.demo.domo.com/page/-100003/kpis/details/223631403
  - ❏ Retrieve and print all of their names and squads.
  - ❏ Update the dictionary by moving one engineer to a new squad.
  - ❏ Print out the new dictionary.
- ❏ Create a dictionary called countries.
  - ❏ Add to the dictionary keys: United States, Argentina, Greece, China, Italy, Canada.
  - ❏ Print out the dictionary.
  - ❏ Add the dates when the countries were established:
    - ❏ United States: July 4, 1776
    - ❏ Greece: February 3, 1830
    - ❏ Argentina: May 25, 1810
    - ❏ China: October 1, 1949
    - ❏ Italy: March 17, 1861
    - ❏ Canada: July 1, 1867
  - ❏ Print out the new dictionary
  - ❏ Add to your dictionary the capital cities of each country:
    - ❏ United States: Washington D.C.
    - ❏ Greece: Athens
    - ❏ Argentina: Buenos Aires
    - ❏ China: Beijing
    - ❏ Italy: Rome
    - ❏ Canada: Ottawa
  - ❏ Print out the country name, founded date, and capital for Argentina.

- ❏ Add an additional country of your choosing to the dictionary.
- ❏ Print out the final contents of the dictionary

Quiz:

When should you use a dictionary instead of a list?
What are the differences between a dictionary and a list?

Demos:

What did you do? Show the completed program.
How did you do it? Walk through the code.

Chapter 5: Conditionals, Loops, and Some Other Statements

Topics:

Printing, Importing, Assignments, Blocks, Conditionals, Assertions, Loops, List comprehension

Tasks:

- ❏ Complete Reading from Beginning Python Chapter 5
- ❏ Given a score between 0 and 100, print out a student's letter grade. Use if statements to determine the grade.
- ❏ Given an array of student scores, such as scores = [76,57,98,84,68,93,99,100] use a for loop to iterate over the array and print out the letter grade of each score.

Projects:

- ❏ Create a new Selenium Project in PyCharm
  You will need ChromeDriver: http://chromedriver.chromium.org/ This will also need to be added to your PATH. An example Python script and instructions are located here: http://chromedriver.chromium.org/getting-started
- ❏ Create Your first Selenium script:
  - ❏ Open a Browser.
  - ❏ Open a URL of your choice.
  - ❏ Click on some element on the screen.
  - ❏ Close the browser.

Quiz:

1. What command is used in selenium to send text to an input box?
2. What is a list comprehension?

Demos:

Demo your automation.
How did you click elements on the screen?
For the student scores assignment can you think of a better way to get grades than an if statement?

Chapter 6: Abstraction

Topics:

Abstraction
Functions
Parameters
Scopes
Recursion
Functional Programming

Tasks:

- ❏ Complete Reading from Beginning Python Chapter 6
- ❏ Create a function that will calculate a student's grade:
  - ❏ Get up to 10 score grades I.e 0-100 score from a user
  - ❏ The function will then calculate the average score and print a letter grade.
  - ❏ Yes use previous chapter tasks to accomplish this.
- ❏ Write a function called recursion that takes as a parameter a number provided by a user and then adds all numbers up to the number given and outputs the total. (Number provided should be between 0 and 100) i.e. recursion(8) outputs 36

Quiz:

1. Given the following code, what is the required input and what is the output:

```
import sys
import random

ans = True

while ans:
   question = input("Ask the magic 8 ball a question: (press enter to quit) ")
   answers = random.randint(1, 8)

   if question == "":
       sys.exit()
   elif answers == 1:
       print("It is certain")
   elif answers == 2:
       print("Outlook good")
   elif answers == 3:
       print("You may rely on it")
   elif answers == 4:
       print("Ask again later")
   elif answers == 5:
       print("Concentrate and ask again")
   elif answers == 6:
       print("Reply hazy, try again")
   elif answers == 7:
       print("My reply is no")
   elif answers == 8:
       print("My sources say no")
```

2. Given the following code, what is the required input and what is the output:

```
total = 0.0
count = 0
while count < 3:
   number = float(input("Enter a number: "))
   count += 1
   total += number
average = total / 3
print ("The average is ", average)
```

Chapter 6: More Homework

Tasks:

- ❏ Create a function called hello that prints out “Hello World!”
- ❏ Create a function that takes a parameter and prints out the parameter to the screen.
- ❏ Create a function that has a parameter with a default value and prints out a message using the value of the parameter to the screen. Test by passing in a value and not passing in a value .
- ❏ Create a function that takes in a number and returns the number multiplied by itself (number squared). Print the returned value to the screen.
- ❏ You are given this code:

```
def list_benefits():
    pass
def build_sentence(benefit):
    pass
def name_the_benefits_of_functions():
    list_of_benefits = list_benefits()
    for benefit in list_of_benefits:
        print(build_sentence(benefit))

name_the_benefits_of_functions()
```

1. Modify the function named list_benefits() to return the following list of strings: "More organized code", "More readable code", "Easier code reuse", "Allowing programmers to share and connect code together"
2. Modify the function named build_sentence(benefit) which receives a single argument containing a string and returns a sentence starting with the given string and ending with the string " is a benefit of functions!"
3. Your output when done modifying the above functions should look like:

```
More organized code is a benefit of functions!
More readable code is a benefit of functions!
Easier code reuse is a benefit of functions!
Allowing programmers to share and connect code
together is a benefit of functions!
```

Chapter 7: More Abstraction

Topics:

Classes
Objects
Polymorphism
Encapsulation
Inheritance
Interfaces and introspection
Object-oriented design

Tasks:

- ❏ Complete Reading from Beginning Python Chapter 7
- ❏ Create a class called duck.
    - ❏ Your duck should be able to speak. (print quack quack to the screen)
    - ❏ Your duck should be able to walk. (print waddle waddle to the screen)
    - ❏ Your duck will also be tracked to show how far it has walked. (each time walk is called the distance traveled will go up by 5 feet, and the total distance will be printed to the screen)
    - ❏ Your duck can be fed. If you feed your duck corn or lettuce it will say, “Yum yum!” if you feed it anything else the duck will say, “Yuck!”

Quiz:

What is pseudocode?

Demos:

What did you do? Show the completed program.
How did you do it? Walk through the code.

Chapter 7: More Classes

Tasks:

- ❏ Create a class called Animal
    - ❏ Your animal can speak. (Be creative with what this means.)
    - ❏ Your animal can move. (Again pick what this means to you i.e. fly, walk, run, swim)
    - ❏ Your animal can see. (When your animal "sees" then it will tell you what it sees. "I spy…"
    - ❏ Your animal has an age and color. When you get the age or color it will be printed to the screen.

Example run through program:

```
>>>gater = Animal()
>>>gater.setName("Alison")
>>>gater.getName()
>>>My name is Alison.
>>>gater.move()
>>>I waddled 5 steps.
>>>gater.see()
>>>"I spy something blue."
>>>gater.see()
>>>"I spy a little duck."
>>>gater.setAge(10)
>>>gater.getAge()
>>>"I am 10 years old."
>>>gater.setColor("green")
>>>gater.getColor()
>>>"I am green and really big!"
>>>gater.setColor("brown")
>>>"I am brown and really big!"
```

- ❏ Very similar to previous Animal class, but this time you will use parameters:
- ❏ Create a class called Animal that will take as parameters: name, age, color
- ❏ Now run the same commands above, but this time don't use the setName, setAge, or setColor calls. (You should just be able to do gater.getName() etc.)

- ❏ Subclass Animal by creating two new classes called Dog and Cat.
- ❏ The Dog and Cat should not say, "I am color and really big" instead they should say something different. Also, they shouldn't "waddle" they should do something different,

like a cat or dog would do. However, they should use the other functions from Animal instead of creating their own.

Demo:

What did you do? Show the completed program.
How did you do it? Walk through the code.

Chapter 7: Multiple Files

In this project we are starting to bring everything together. The basic requirements are to review how to create a new project in PyCharm, create a class file, and create an additional file that uses the class file for the program.

You will be starting up a new library of classical books:

> ***CREATE A NEW PROJECT IN PYCHARM***
> ***Create a file for your Library class***
> Your Library will have a set of books (dictionary). Each book will have a number for the ID of the book and will also have the title, author, pages and published date. Each of these will be strings to start except for pages which will be a number i.e. int.
> Add 3 methods to your class:
>
> > addBook:
> >
> > > Take 4 parameters: Title, Author, Pages, Publication Date
> > > Adds the book to the dictionary with a new # for the ID of the book
> >
> > printBooks
> >
> > > Prints all the books
> >
> > printBooksByAuthor
> >
> > > Allow user to search for a book by author name, if the name matches an
>
> author in the Library, then print out the details of the book.
>
> ***IN A SEPARATE FILE BUT IN THE SAME PROJECT***
> Create a program to use your Library

1. Print out all the books in your library
2. Ask the user for the necessary information to add a book to your library (Add the book to the library. :-))
3. Ask the user to search for a book by author
    a. Find the book and print it to the screen.

Some books you can use:
title: 'The Adventures of Huckleberry Finn'
author: Mark Twain
pages: 102
publishedDate: 'December 10, 1984'

title: 'Frankenstein'
author: 'Mary Shelley'
pages: 280
publishedDate: 'January 1, 1818'

title: 'Pride and Prejudice'
author: Jane Austen

pages: 432
publishedDate: 'January 28, 1818'

title: "Great Expectations'
author: 'Charles Dickens'
pages: 544
publishedDate: '1861'

title: 'The Scarlet Letter'
author: 'Nathaniel Hawthorne'
pages: 206
publishedDate: 'March 1, 1850'

Demo:

What did you do? Show the completed program.
How did you do it? Walk through the code.

Real World Project:
Create a random salary list of 10,000 salaries using Python to generate the numbers and then export them to a file.
File should look something like this:

```
$105316
$128255
$55212
$45897
$47550
$156010
$82076
...
```

Bonus Points:
Add the $ sign before each salary.

```
$105316
$128255
$55212
$45897
$47550
$156010
$82076
...
```

Format the number with commas like a real dollar value:

```
$105,316
$128,255
$55,212
$45,897
$47,550
$156,010
$82,076
...
```

Chapter 8: Exceptions

Topics:

Exception Objects
Warnings
Raising Exceptions
Custom exception classes
Catching exceptions
Else clauses
Finally
Exceptions and functions

Tasks:

- ❏ Complete Reading from Beginning Python Chapter 8
- ❏ In Chapter 5 you wrote a function to calculate a student's grade. Add exception handling to this function. Raise exceptions if a number between 0 and 110 is not passed to the function.
- ❏ Write a new function that takes 2 numbers and outputs:
    - ❏ The Sum, Product, Dividend, and Difference of the two numbers
    - ❏ Add exception handling. If the difference is less than 0 raise a human readable error message. If something other than a number is passed to the function throw an error that says, "Silly human, you must pass in a number!"