ZimCode 12 Week Curriculum Exercises

These are exercises for each lesson based on the lesson plan attached that follows the textbook content. Try to do as many of the exercises as possible. Solutions are provided for some of the problems but very few of them. If you don’t know the answer to a problem don’t skip it! Ask!

**You can add solutions to the problems but if you want to change the problem, please do it by commenting.**

# 0. Introduction to Computers

## 0.1: Master Hacker

Use the following commands on the command line (Command Prompt) to put a new text file called “myfile.txt” in a folder you make in “My Documents”

* 1. **cd** into drive C:
  2. use **dir** to check if there is a folder called something like “users”
  3. **cd** into the users folder
  4. use **dir** to see the name of the user folder (something like ZimCode) and **cd** into it
  5. make a new folder called “MyText” using **mkdir**
  6. **cd** into the folder
  7. Use the command **notepad myfile.txt** to create a new text file and save it.
  8. Use the GUI browser (Windows Explorer) to see if the file is where you put it.

Solution

This should be after file systems, files, folders, drives etc. are explained.

> cd C:

> dir

> cd users

> cd zimcode

> mkdir MyText

> cd MyText

> notepad myFile.txt

# 1. Introduction to python

## 1.1: How to Print

1. Use the print function to print the words “No Syntax Error”.
2. Now try printing it using the following lines of code. Look for what is causing an error and read the error that is produced in each case

>>> pint(“No Syntax Error”) #Why is this an error?

>>> print(No Syntax Error”) #Why is this an error?

>>> print(“No Syntax Error’) #Why is this an error?

>>> print(“No Syntax Error” #Why is this an error?

>>> print(“No “Syntax” Error”) #Why is this an error?

>>> Print(“No Syntax Error”) #Why is this an error?

All these errors are called syntax errors which happen when you type things the computer does not understand. (It’s like the sounds a baby is making, you can hear them but they don’t make sense because they are not in a correct language) It’s useful to remember what the error says so that you know which mistake you made.

## 1.2: Adding Machine

*Ask the students for the answer then show them.*

Calculate the following quantities by head/using a calculator and then using Python

* 1. 48\*3
  2. 9/7
  3. 5-4
  4. 0-7
  5. 0.0004+0.0005
  6. 2\*2\*2\*2\*2
  7. 2\*(90-3)/72+17\*0.5
  8. 2\*3-1
  9. 2\*(3-1)
  10. 2(2) #Why is this an error?

## 1.3: Square Dancing

Calculate the following quantities by head/using a calculator and then using Python

1. 2\*\*2
2. 2\*\*5
3. 0.1\*\*3
4. 10\*\*0
5. 10\*\*1000
6. 100\*\*0.5
7. 81\*\*(1/3)
8. (12.6\*(90-8)/33)\*\*0.5
9. (1+1/100+(1/100)\*\*2+(1/100)\*\*3)\*\*2
10. 2\*\*\*2 #Why is this an error?

## 1.4: Carpentry 101: Modding Floors

1. 13//4
2. 13//5
3. 13//3
4. 5//1
5. 14\\6 #Why is this an error?
6. 5%2 #Can you use this to check if a number is divisible by 2?
7. 23673%345
8. (34%7)//3
9. 34%23//3
10. 34%(23//3)

## 1.5 Maximum Power!

1. pow(2,2)
2. min(1,2,3,4,5)
3. max(1,2,3,4,5)
4. min(2\*\*0,2\*\*1,2\*\*2,2\*\*3)
5. max(min(1,2,3,4,5),min(6,7,8,9,10))
6. sum(1,1,1,1,1)
7. sum(0,0,0,0,43)
8. max(1.2.3.4.5) #Why is this an error?
9. max(01,2,3,4,5) #Why is this an error?
10. min(max(1,2) #Why is this an error?

## 1.6: Which country do we import our math from?

1. Import the math module into the shell (**import math**)
2. Use **math.cos()** to find cos(3.14)
3. Use **math.pi()** to find the area of a circle of radius 3
4. Use **math.log()** to find the natural logarithm(ln) of 2.72
5. Use **math.atan()** to find 180/3.14\*tan-1(1/(3\*\*0.5)
6. Use the following commands and find the error

>>> ath.cos(3)

>>> math.cost(3)

>>> math,cos(3)

>>> math.Cos(3)

# 2. Variables

## 2.1: Independent variables

Declare/Define the following variables:

1. x = 2
2. y = 4
3. myVar = “rent”
4. my\_str = “string”
5. my\_float = 2.3
6. 1\_variable #Why is this an error?
7. my var = 3 #Why is this an error?
8. for = 4 #Why is this an error?
9. a = #4 #Why is this an error?
10. var = 12 #Why is this an error?

## 2.2: Easy Change

1. Declare/Define the following variables
   1. x = 3
   2. y = 5
   3. z = 4.5
2. print the variable x
3. print the variable z
4. print the solution to x+z
5. print the solution to x+y+z
6. print the solution to x\*\*y/(y//z)%(2+x)

## 2.3: Garbage in Garbage Out

1. Use the **input()** function to get the user’s name.

input(“What is your name”)

1. Use the **input()** function to get the user’s name and store the name in a variable called **user\_name.**

name = input(“What is your name?: ”)

1. Use the **input()** function to get the user’s name and print it.

name = input(“What is your name”)

print(name)

or

print(input(“What is your name?: ”))

1. Ask different questions like what is your age, school, address etc. and print them. Take note of any errors.

## 2.4: Scripts

1. Create a new script and save it in “My Documents”, name it whatever you want.
2. Delete the script you just created.
3. Create a new folder in My Documents and call it name-school where you put your name where there is name and your school where there is school. If I am a Bothabo from Mzilikazi, it will be Bothabo-Mzilikazi.
4. From now on, every script you write should be saved in this folder with a different name. Make sure to use the same computer

## 2.5: Taking Comments

1. Write a new script with the date as a comment on the first line
2. Use the script to write a short program that asks the user for the date and then prints “Hello World”
3. Comment out the line where you print ”Hello World” and add a new line where you print the date that the user inputs in step ii.
4. Now comment out the line where you get input from the user. Run the script. Why is this an error?

#27/05/18

date = input(“What is the date?: “)

#print(“Hello World”)

print(date)

# 3. Data Types

## 3.1: Do you think you’re my type?

What are the following types? If you are not sure, use the type function.

|  |  |
| --- | --- |
| **Value** | **Solution** |
| 1. 234523 | int |
| 1. 234.42 | float |
| 1. “ere” | string |
| 1. ‘34’ | string |
| 1. None | None |
| 1. 34/23 | float |
| 1. 34//23 | int |
| 1. (7\*6)/2 | int |
| 1. ’35.5’ | float |
| 1. ‘None’ | string |
| 1. False | Boolean |
| 1. ‘false’ | string |

## 3.2: Cast away

Identify the type of the following values and then cast them into the type shown. If they cannot be cast, explain why.

1. 15 – float
2. 1.5 – str
3. “23” – int
4. “34.5” – int
5. “True” – bool
6. 23 – int
7. 45 – str

## 3.3: Stop Boolean me

First solve these by head then try out the answer in the Python Shell

1. 1>2
2. 2\*3 < 3\*2
3. 12 < 45
4. 12 < 4.5
5. 2 <= 56
6. 2 <= 2
7. 45 == 4.5
8. 87 >= 32
9. 34 != 12
10. 23 != 23
11. 3.0 == 3
12. 3.01 < 45 > 23
13. ‘3’ == 3
14. ‘ya’ == ‘yes’
15. ‘yes’ != ‘no’
16. 4.5 == 4.50
17. 3>2 and 2<32
18. 0 == False #False is actually just the integer 0 in Python
19. 0 == True
20. 1 == True #True is actually just any integer except 0 in Python
21. “yes” == “yes” and “no” == “no”
22. 1 == 1 or 2 == 3
23. 1 != 17 or 34 == 34
24. 12 <= 11 or “hello” == “hi”
25. (1 > 3 or 3 <45) and (23 >=21 or 54 < 34%4)
26. 1 is 1 #This works but always use == in this class.
27. 2 is not 3
28. not True
29. 1 or 0
30. 0 or 1
31. not 3 > 4
32. ‘a’ in ‘animal’
33. ‘b’ not in ‘zithini’
34. ‘ola’ in ‘ola mfethu’
35. ‘etha’ not in ‘ndeipi’
36. 2 = 3 #Why is this an error?
37. 34 =! 34 #Why is this an error?
38. 23 isn’t 3 #Why is this an error?
39. yes == 1 #Why is this an error?

## 3.4: Stringing Me Along

All of the following strings have an error, find it and explain why:

1. “My string’
2. Why should I follow?
3. “He thinks he’s “funny”, doesn’t he?”
4. ‘Listen to me

## 3.5: The length of my string

1. Find the length of the string “Operate” using len(“Operate”)
2. Find the length of the string “Fun Times” using len(“Fun TImes”)
3. Find the length of the string “ ” using len(“ ”)
4. Use **str()** to convert the integer 234 into a string.
5. Use **str()** to convert the float 45.8 into a string.
6. Declare a variable called **first\_name** that stores your first name as a string and find its length
7. Declare a variable called **school** that stores your school as a string and find its length
8. Type cast the integer 1234 into a string using str(1234)
9. Type cast the float 65.4 into a string using str(65.4)
10. Import the math module, convert the value of pi from math.pi into a string and print it

import math

print(str(math.pi))

#or

pi\_str = str(math.pi)

print(pi\_str)

1. Try calling the following functions and explain why there is an error
   1. length(“Hello”)
   2. len(“Hello)
   3. len(Hello)
   4. len(1234)
   5. len(‘1234’
   6. str(my string)

## 3.6: What is Concatenation?

1. Join the two strings “Hello” and “World” using he line “Hello”+”World”
2. Join the string “123” with “456”
3. Define the variable **first\_name** as your first name, **last\_name** as your last name and join them. Make sure there is a space in between them.
4. Define the variable **num** as your favorite number, type cast into a string, join it at the end of a sentence that says “My favorite number is “ and print the whole thing on one line.

num = str(5)

print(“My favourite number is “+num)

1. Use input() to get the user’s name and then the computer should reply “Hello *name”* with *name* replaced by the user’s name

name = input(“What is your name?: “)

print(“Hello “+name)

1. Write a program that gets the name, age and school of the user and stores them all in different variables then makes the computer print a sentence saying “Hi *name,* you are *age* years old and you go to *school*” where you put the *name, age* and *school* in the right place.

name = input(“What is your name?: “)

age = input(“What is your age?: “)

school = input(“What is your school?: “)

print(“Hi “ +name+”*,* you are “ +age+ “years old and you go to “+ school)

1. Why do the following lines of code give you errors?
   1. ‘hello’ + world
   2. “this is” + 3
   3. len(“word”+ 4)

## 3.7: Indexing

1. At what index(es) (position) in Python is the character “a” in the following strings:
   1. “abcde”
   2. “hat”
   3. “fearsome”
   4. ‘Python is a fun language’
   5. ‘Hahahaha’
2. What is the slice that contains the string “hat” in the following strings

|  |  |
| --- | --- |
| **String** | **Solution** |
| * 1. “My hat” | 3:6 or 3: or -3: |
| * 1. “hatsarecool” | 0:3 or :3 |
| * 1. “Your hat” | 5:8 or 5: |
| * 1. “shatter” | 1:4 |
| * 1. “that” | 1: or 1:4 |

1. What is the slice that contains the string “sing”? If it backwards, reverse it so that it is the correct direction

|  |  |
| --- | --- |
| **String** | **Solution** |
| * 1. “sing” | : or 0:5 |
| * 1. “gnis” | ::-1 or 0:5:-1 |
| * 1. “I sing” | 2: or 2:6 |
| * 1. “Do you gnis bra?” | 7:11:-1 or -8:-4:-1 |
| * 1. “Ignis Verblund” | 1:5:-1 |

## 3.8: String methods

Put the string where there is “s”, do each example multiple times with different words, testing the students to see if they understand what each one does.

1. Change “asa” to uppercase using s.upper()
2. Change “EVERYDAY” to lowercase using s.lower()
3. Capitalize the first character of “thursday” using s.capitalize()
4. Find the index of a in “Yeah” using s.index()
5. Check if “isitshwala” is lowercase using s.islower()
6. Check if “Monday” is in upper case using s.isupper()
7. Count how many ‘g’s are in ‘string’ using s.count(“g”)
8. In the shell, enter help(str), this gives you a list of all existing string methods! Look at the list and read what they do.
9. Why is it that s is not capitalized if you type this code in a script? How do you fix it?

s = “naMamo”

s.capitalize()

print(s)

1. Why do the following calls produce errors?
   1. “Hello”.isUpper()
   2. name.lower()
   3. “tuesday”.capitalize
   4. “run”!.upper()
   5. “abcde”.index(“z”)

# 4. Functions

Notice the difference between prints and returns. As the tutor, **always write doc strings** so that the student can read in words what the function does.

## 4.1: Maths futhi

1. Define a function called f(x) that prints there answer to f(x) = 2\*x+1 e.g. f(1) = 3.
2. Define a function called g(x,y) that prints the answer to g(x,y) = x\*\*2+y\*\*2 e.g. g(1,2) = 5.
3. Define a function called rect\_area(length, width) that returns the area of a rectangle given by length\*width e.g. rect\_area(3,4) returns 12

## 4.2: How do functions function?

1. Define a function called hello() that prints the word “Hello” (No parameters).
2. Define a function called subtract(x,y) that subtracts y from x and prints the answer. e.g. subtract(4,2) prints 2
3. Define a function called concatenate(str1, str2) that concatenates str1 and str2 and returns the answer then print it. e.g. concatenate(“hello “,”world”) returns “hello world”.
4. Define a function called full\_stop(s)that takes any string s and and puts a period (full\_stop) at the end. e.g. full\_stop(“My sentence”) returns “My sentence.”
5. Define a function called sentence(s) that capitalizes the first letter of s and adds a period at the end e.g. sentence(“short sentence”) returns “Short sentence.”
6. Define a function called circle\_area(radius) that finds the area of a circle of radius given by radius using math.pi from the math module. e.g. circle\_area(1) = 3.1415936
7. Define a function called divisible\_by\_2(num) that tests if num is divisible by 2 and returns a boolean e.g. divisible\_by\_2(4) returns True.
8. Define a function called cel\_to\_fahrenheit(temp) that converts temperature temp in Celsius to Fahrenheit where TCelsius= TFahrenheit\*9/5+32 (Yes, let them do the maths, this give Fahrenheit to Celsius but we want Celsius to Fahrenheit)
9. Why is it that this script does not print anything in the shell? How do you make it print?

def my\_func():

return “Ayaya”

1. Why do the following function definitions give errors?

#1

def f(x)

return x+1

#2

def f():

print(“hello”)

#3

def 1\_func():

return 1

#4

def add two(x):

return x+2

#5

def all\_caps(s):

s = s.capitalize()

return s

## 4.3: But I did programming to avoid compositions!

1. Do all of the following things in one line of code:
   1. convert the string “ZVAITA” to lowercase then print it
   2. print the length of the string “Long string”
   3. print the maximum number from the set 1,2,3,4,5,6
   4. Cast the integer 1 into a string, concatenate the string “Number “ to the beginning and then print the result. e.g. should print “Number 1”.
2. Use the input() function to get a number from the user, then cast it into a float and print it in one line.
3. Use the input() function to get a number from the user, add 5 to it and then print the solution in one line. e.g. if the user enters 4.3 it should print 9.3.
4. Write a function called greet() that asks the user for their name then prints the statement “Hello *name”* where *name* is the users name. It should have only one line of function code.
5. Why do the following lines of code give errors? Fix them.

#1

print(input(“What is your name?: “)

#2

print(int(“3”)+”2”)

#3

len(str(“Agent 00”+7))

#4

Print(“The answer is “+str(float(“3.4”)+1.2)

## 4.4: Putting it all together

These are slightly longer examples that combine all the things learned so far.

1. Define a function called divide() that takes two integers from the user and prints the statement “*x* divided by *y* is *x//y* remainder *x%y”* where you replace x and y with numbers. e.g. divide(5,3) should print “5 divided by 3 is 1 remainder 2”.
2. Define a function called Pythagoras(a,b) that returns the hypotenuse side of a right angled triangle using Pythagoras’ theorem (c\*\*2 = a\*\*2 + b\*\*2). Use this in a program that asks for input from the user and prints the result.
3. You are hired by Zim Bank to create a mobile app for their banking services and you are on the team that does loans and savings. Define a function that will calculate the compound interest for any amount they input. Remember that compound interest is calculated by

where P is the principal, r is the decimal interest rate (e.g. r=0.05 is 5% interest per annum) and t is the number of years.

1. Write a program that allows the user to write their own short story using a template you provide. It should be a minimum 50 words and the user must enter input at least 6 times. It should first print the template and show what gaps should be filled then allow the user to enter words to fill in the gaps. Give an example then allow the students to write their own in whatever language they want. To use more than one line, press enter without putting a closing parenthesis and idle will put you in the correct indentation.

print(“This was a shock, a far greater shock than \_\_\_\_(1) could ever imagine. How could \_\_\_\_(2) do that. They couldn’t even remember the last time they \_\_\_\_\_(3). Something had to be done and that thing was \_\_\_\_(4). If only there was \_\_\_\_(5). The result will still be the same, but at least \_\_\_\_\_(6) would have tried to \_\_\_\_(7).”)

str1 = input(“Enter phrase for (1): ”)

str2 = input(“Enter phrase for (2): ”)

str3 = input(“Enter phrase for (3): ”)

str4 = input(“Enter phrase for (4): ”)

str5 = input(“Enter phrase for (5): ”)

str6 = input(“Enter phrase for (6): ”)

str7 = input(“Enter phrase for (7): ”)

print(“This was a shock, a far greater shock than”,str1,”could ever imagine. How could”,str2,”do that. They couldn’t even remember the last time they”,str3,”. Something had to be done and that thing was”,str4,”. If only there was”,str5,”. The result will still be the same, but at least”,str6,”would have tried to”,str7,”.”)

1. Define a function called format\_str(s) that takes in a string s and prints the string in the following formats on each line: all lowercase, all uppercase and capitalized. e.g. format\_str(“mY String”) gives:

>>> my string

>>> MY STRING

>>> My string

# 5. Flow Control

Do this topic at the pace of the class, the slides are for 2 lessons but if you need one more it is fine. It does not matter if the different schools end at different topics. Remember that wherever you stop in the slides, the next time you must start by reviewing the previous lesson to remind people what is going on.

## 5.0: Review exercise 3.3

## 5.1: So Random

1. Import the random module
2. Use the function random.random() to print a random float between 0 and 1
3. Use the function random.randint(start,stop) to print a random integer between 1 and 100
4. Use the function random.random() to produce a random integer between 1 and 100

import random

num = int(random.random()\*100)

#or

num = random.random()\*100//1

print(num)

1. Use random.choice(s) to pick a random character in the string “Random String”
2. Repeat i-v using different numbers

## 5.2: What if my input is this? (Short Exercises)

**Remember to always write doc strings as a tutor!**

1. Write a program that prints “Yes” if 5 is greater than 4.
2. Write a program that prints “Too long!” if the length of the string “supercalifragilisticexpialidocious” is greater than 15. Fun Fact: It’s the longest word in the English dictionary!
3. Define a function called more\_than\_5() that asks for a word from the user, if the length of the word is more than 5 letters, it should print “It is more than 5”
4. Define a function called divisible\_by\_2(n) that returns True if a number is divisible by 2 else returns false. Define one that checks divisibility by 3,4,5 etc.
5. Define a function called is\_first\_upper(s) that returns True if the first character of the string s is in uppercase
6. Define a function called is\_a\_in(s) that returns True if the character “a” is in the string s otherwise returns False.
7. Define a function called is\_char\_in(s,char) that returns True if char is in the string s otherwise returns False.
8. Find the error in the following code blocks

#1

x = 4

if x > 5

print(“x is greater than 5”)

#2

y = “up or down”

if “up” in y:

print(“up!”)

#3

z = 34

if z%3 != 4:

print(“Eish”)

else:

print(“Yah”)

#4 – Why does this always print “Always the case”?

if True:

print(“Always the case”)

else:

print(“Never happens”)

#5

num = input(“Enter a number”)

if num > 5:

print(“It is greater than 5”)

**This is the point in the course when things start becoming more practical and interesting. Many of the following examples will come with a backstory to help you and the students see how this can be applied in the real world. Please include the story when describing the problem.**

## 5.3: So Many Options

1. Write a program that asks for a colour from the user (“black” or “white”). Then:

* If it is black, print “You chose black”
* Else If it is white, print “You chose white”
* Else print “Please choose black or white”

You can add more colours to the list (more elif statements)

1. A lady at a msika asks you to program a swipe machine for her msika. Write a program that asks the user to enter how much money they have then define a variable called **cost** andset it to whatever amount you want (e.g. 10) then:
   * If the user has less money than **cost** print “You do not have enough money in your account”
   * If the user has money that is greater than or equal to **cost**, then print “Transaction successful” and outside the if statement print how much money they have left.
2. You are designing a self-driving car and you want to make a car that follows the rules at a robot. Write a program that asks for the colour of the robot from the user then define a function called **robot\_func(colour)** that takes in the colour entered by the user and
   * If it is green, print “go”.
   * else if it is amber, “slow down and go if safe”
   * else, “stop”
3. Design a rock, paper, scissors game for two people. Ask for input from player\_1 and input from player\_2 (they can see each other’s answers, it doesn’t matter. The players can either put in:
   * ‘r’ for rock
   * ‘s’ for scissors
   * ‘p’ for paper

Now write, if,elif, else statements to find the winner e.g.

if player\_1 is rock and player\_2 is scissors, print “Player 1 one wins”. Hint: There are many ways to do this, you can use the **and** keyword (9 cases). e.g.

* + if player\_1 is paper and player\_2 is scissors, print “Player 2 wins”

1. Why is it that the following code blocks will never print the words “Never happens” or has an error.

#1

inp = float(input(“Enter a number: ”))

if inp > 5:

print(“Greater than 5”)

else:

print(“Equal to 5”)

elif inp < 5:

print(“Less than 5”)

#2

inp = float(input(“Enter a number: ”))

if inp > 5:

print(“Greater than 5”)

elif inp > 5:

print(“Never happens”)

else:

print(“Equal to 5”)

## 5.4: Bird’s nests

1. You are an ecologist (someone who studies nature and animals) and you want to take pictures of birds in their nest but if you stand close to the nest, the bird will fly away so you decide to program a hidden camera to take the picture automatically. Your camera has 2 settings: night mode (for when it’s dark) and normal mode. Write the if statements (not in code, in English) for taking pictures anytime during the day. You also have sensors that tell you if the bird is there.

#Answer to i.

#if (It is night):  
 #if (bird is there):

#Take pictures

#else:

#Do nothing

#else:

#if (bird is there):

#Take pictures

#else:

#Do nothing

1. Write a program that stores a variable x and tests the following conditions, replace the conditions and instructions with actual code.

#if (x is greater than 10):

#print that x is greater than 10

#if (x is divisible by 2):

#print that x is greater than 10 and divisible by 2

#else:

#print that x is greater than 10 bu not divisible by 2

1. Define a function called is\_capitalized(s) that returns True if a word is capitalized. The if statement branches are as follows, replace the conditions and instructions with actual code. (Hint: You are writing if statements inside other if statements inside functions, indentation is important, make sure students understand indentation)

#if (first character in s is upper case):

#if (everything else is lower case):

#return True

#else for all other cases, return False

(Be careful if you use else statements and make sure they cover all cases)

# 6: Loops

Do this topic at the pace of the class, the slides are for 2 lessons but if you need one more it is fine. It does not matter if the different schools end at different topics. Remember that wherever you stop in the slides, the next time you must start by reviewing the previous lesson to remind people what is going on.

**Do every single example problem. Yes there are lots of them but as you know very well, loops are hard to understand so we will just beat it into them. This might take 1 or 2 extra lessons, the syllabus allows this, it’s fine.**

## 6.1: Slow as a turtle

Background: px means pixels. A pixel is one block of colour on the screen. A screen with 1080p has 1080 pixels from left to right. The more pixels you have, the better the resolution of the screen. Resolution is the ability tell that two that are close together are not the same.

1. Write a short program that draws a square of side 200px using turtle
2. Write a short program that draws a rectangle of side 100x20px
3. Write a short program that draws a square and a rectangle on the same screen but they should not touch (Hint: Use turtle.penup()).
4. Use turtle.circle(radius) to draw a circle.
5. Why does the following code give an error?

#1 – You need to import turtle!

turtle.forward(100)

#2 – It’s turtle.forward(100) not forward(100)

import turtle

forward(100)

#3 – Always check spellings!

import turtle

turtle.Forward(100)

## 6.2: This will take a while

1. Use a while loop to print the words “Eish” 10 times. **Do this example with and without a break statement**

#Solution with break statement

n = 0

while True:

if n >= 10:

break

print(“Eish”)

n += 1

1. Use a while loop to print the words “Forever on the screen. (To stop it, press ctrl+C in the shell, this is called a keyboard interrupt.)
2. Use a while loop to print the numbers in the following arithmetic sequences given below:
   1. 1,2,3,4,5
   2. 2,4,6,8,10,12
   3. 100,150,200,250,300
3. Use a while loop to print the following sequences (you should print the answer):
   1. 20, 21, 22, 23, 24, 25  should print the geometric series (1,2,4,8,16,32)
   2. 20, 22, 24, 26, 28, 210
   3. log10(11), log10(12), log10(13), log10(14) (Hint : Use math.log10(x))
4. Use a while loop to calculate a running sum for the sequence in (iii. a.) and print at the end (only at the end! – what does this tell you about where to indent the print() command?) i.e. it should find 1+2+3+4+5. Make the sequence go up to 34. How much code would it take to do this manually?
5. Generate a random number integer between 0 and 100 and store it in a variable **num** and print it(Remember the random module?). Use a while loop to print the number of times the loop has run until it reaches the number given by **num**. Example below:

>>> num is 5

>>> 1

>>> 2

>>> 3

>>> 4

>>> 5

>>>

1. Use a while loop to keep asking the user for input until they type in the word ‘stop’. When they type in ‘stop’, print the words ‘Now stopped’. See the example below.

>>> Do you want to stop? d

>>> Do you want to stop? Stop

>>> Do you want to stop? No

>>> Do you want to stop? Haibo

>>> Do you want to stop? stop

>>> Now stopped

>>>

1. Why do the following code blocks/snippets produce errors?

#1

while x < 10:

print(x)

x += 1

#2 - Why does this never print anything?

x = 10

while x < 10:

print(x)

x += 1

#3

while True:

print(“Forever”)

#4 – Change this code to make sure 13 the last number included in #the sequence printed

n = 3

while n<13:

print(n)

n+=2

## 6.3: Indexing again and again

1. Review Ex. 3.7
2. Use a while loop and indexing to print every character in the words:
   1. “Hello”
   2. “Many Characters”
3. Use a while loop and indexing to print every character in the following words in upper case:
   1. “Hello”
   2. “Many Characters”
4. Why do the following code blocks/snippets give errors?

#1

i = 0

name = “Bizo”

while i < len(name):

print(name[i])

i += 1

#2 – Why does this run forever i.e. an infinite loop?

i = 0

name = “Bizo”

while i < len(name)-1:

print(name[i])

i += 1

#3

i = 0

name = “Bizo”

while i < len(name)-1:

i += 1

print(name[i])

## 6.4: Shooting Range for programmers

1. What range do you use to produce the numbers 56,57,58,59,60
2. What range do you use to produce the numbers 2,5,8,11
3. What range do you use to produce the numbers 19,17,15,13
4. Do the examples i-v in Ex. 6.2 using for loops
5. Do Ex. 6.3 using for loops
6. For each example in iv and v, is it easier to use a for loop or while loop?
7. Can example 6.2 – vi be done with a for loop? how or why not?

Solution: Can be done with both since user enters the number of times it should be done

1. Can example 6.2 – vii be done with a for loop? how or why not?

Solution: Only while loop works because we never know when we will stop.

1. Why do the following code blocks/snippets produce errors?

#1

print(range(2,-1,1))

#2

for i in range(12)

print(i)

#3

s = “Aya”

for i in range(4):

print(s[i])

#4

s = “Aya”

for i in range(2):

print(s[I])

#5

for i in mystr:

print(i)

#6

mystr = “345”

for i in mystr:

print(i+3)

#7

mystr = “234567”

for i in mystr:

float(i)

print(i+3)

## 6.5: Classic Looping Examples

1. The guessing game – Write a program that produces a random integer between 0 and 100. The user then has to guess the number until they get it correct while the computer gives hints like “Your number is too low”. When the user has guessed the correct number, it should print “Well done, it took you x guesses “ where x is replaced by the number of guesses. So here is the pseudcode:

#Get a random integer

#guess starts False

#while the guess is False

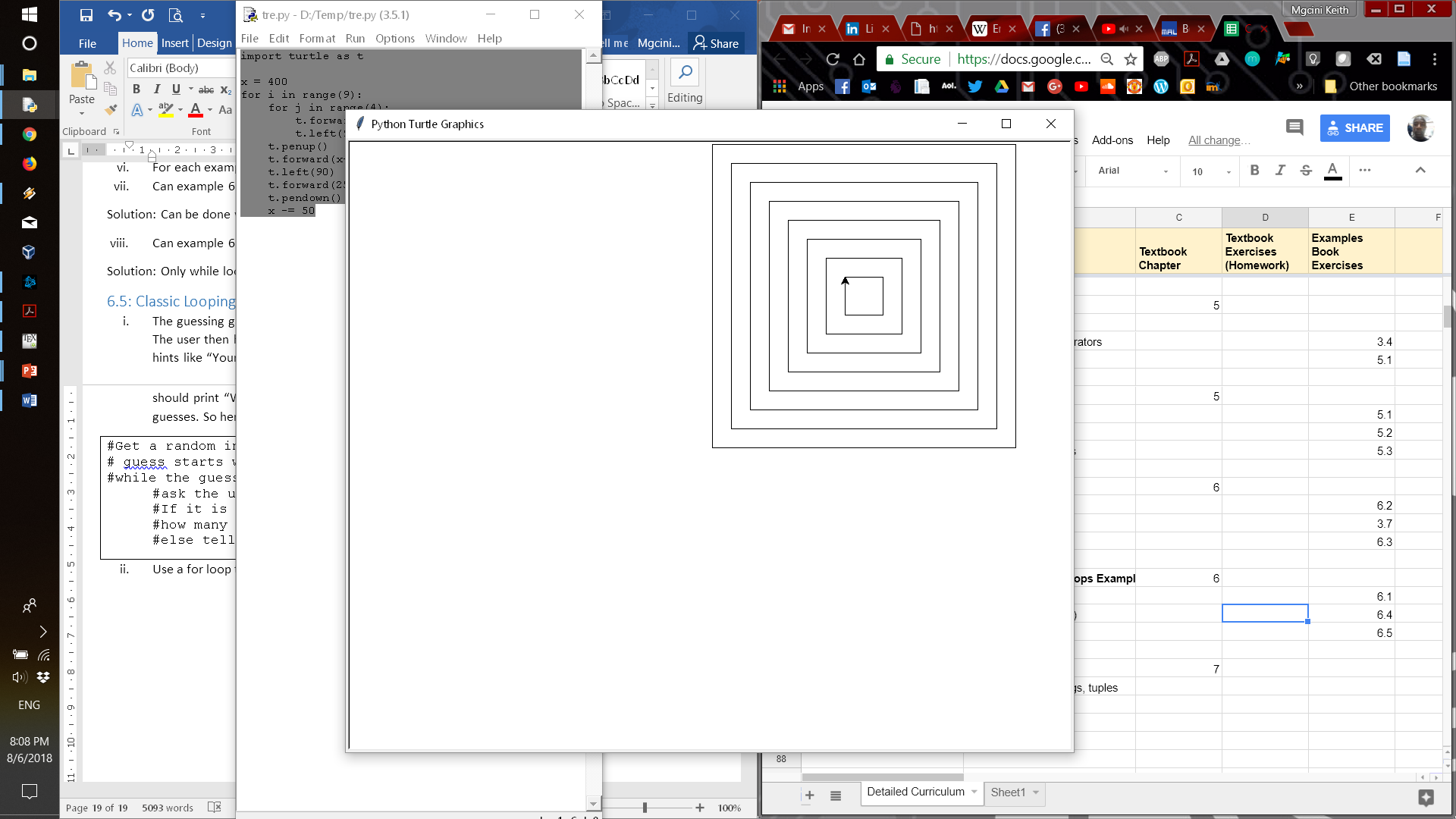
#ask the user to guess

#If it is correct,break out of the while loop and tell them

#how many tries it took

#else tell them if it is too low/high

1. Use a for loop to draw a square in turtle.
2. Use a for loop inside a for loop (nested for loop) to draw the following shape



#solution

import turtle as t

x = 400

for i in range(9): #Draw 9 squares

for j in range(4): #Draw a square of length x

t.forward(x)

t.left(90)

t.penup() #Reduce the size of the next square

t.forward(x-25)

t.left(90)

t.forward(25)

t.pendown()

x -= 50

1. If you were to throw a coin 100 times, how many times do you expect to get heads? We are going to simulate a coin toss using the random module to see how many heads we get. Remember that random.random() from the random module produces a random float between 0 and 1. if we set the variable x to a float created by random.random() let
   1. if x is below 0.5, it is heads
   2. if x is above 0.5, it is tails

Write a program that makes N coin tosses and tells you what fraction of heads it produces. What happens when N is the following values

1. 1
2. 5
3. 100
4. 1000
5. 100000
6. 100000000

Run the script a few times for each value of N. What can you say about the fraction of heads as N increases?

If you do A level maths – What is the distribution for a coin toss? (Solution -> binomial)

Whenever you increase the number of trials of an experiment, your results start to show the real probability of finding heads/tails. (Central Limit Theorem)

import random

N = 1

num\_heads = 0

for i in range(N):

x = random.random() #Produce a random number

if x < 0.5:

num\_heads += 1 #If it is heads increase counter by 1

print(num\_heads/N)

# 7. Data Structures

## 7.1: Born as a string of quintuplets

1. Define a tuple storing 5 names; “Nqo”,”Rudo”,”Joy”,”Abi” and “Bob”. What are the indices of each name in the tuple. Find the length of this tuple.
2. Define a tuple called **ints** that stores the numbers 1,2,3,4,5,6. Find the length of this tuple. Find the sum of all the elements.
3. Revisit indexing 3.7
4. Use the help function to see all the methods defined for tuples.
5. Define the a tuple **t** with elements (34,”test”,True). Using the type function, find the type of
   1. t
   2. the first, second and third element of t
6. From the random module, use the function **random.choice(**ints**)** to print a random number from the tuple in 7.1.ii. random.choice() works on any iterable.
7. Use a for and then a while loop to print all the numbers inside the tuple defined in 7.1.ii
8. Define a function called in\_t(t,val) that returns True if the object val is inside the iterable (in this case tuple t)
9. Why do the following code blocks/snippets give errors

#1

>>> t = (9,10)

>>> t[0] = 10

#2

t = (23,45,43

#3

t = (23.45.43)

#4

t = 74,45,67 #It works without parentheses!

print(length(t))

#5

t = (“1”,”2”,”3”)

print(t[3])

#6

t = (4,5,7)

random.choice(t)

## 7.2: List of Todos

Do different examples of each problem in the shell, change the lists, change the positions etc. l refers to the list on which the method is called.

1. Define a list with 4 numbers of your choice, and find
   1. the length of the list
   2. the sum of the numbers
   3. the type of the list (use the type() function)
   4. The maximum and minimum values of the list
2. Define a function called list\_avg(l) that returns the average(mean) of the numbers in the list l.
3. Use the help function to find the methods that apply to lists.
4. Add the number 5 to the end of the list l = [1,2,3,4]
5. Join the 2 lists [1,2,3] and [4,5] using l.extend() or concatenation (list1+list2)
6. Define a list **l** with **l** = [[1,0],[0,1],[0,1]] i.e. a list of lists and print the element in position 0 of the list in position 1 of **l.** Do this with the other elements
7. Define a list with l = [1,2,33,4,5]. Change the element in position 2 to 3.
8. Define a list with l = [[1,2],[33,4,5]]. Change the 33 to a 3.
9. Define a list with l = [[1,2],[3,4,5]]. Use slicing to print the values 3 and 4.
10. Make shallow copy of the list l = [1,2,3,4,5]
11. Define a function called list\_median(l) that returns the median of the unordered list of numbers l. (Hint: The median is the number in index N/2-1 rounded up if the N is the length of the **sorted** list e.g. the median of [1,2,3,4,5] is 3 and for [1,2,5,67,322,1211] is 67)
12. Use l.index(item) to find the index of 4 in the list [0,1,2,3,4]
13. Use del l[index] to delete the element in position 3 in [1,2,3,4,5,6]
14. Use l.remove(item) to remove the string “bye” from the list [“stay”,”remain”,”bye”]
15. Use l.pop(
16. Why do the following lines of code produce errors

#1

l = [1,2,3)

#2

l1.append(2)

#3

l[4] = 12

#4

l = [2.4,4.5,6.7]

l.append()

#5

l = [0,1,2,3]

del l[4]

#6

l = [“1”,”2”,”3”]

l.remove(1)

## 7.3: Iterating on Lists

**List comprehension is not part of the syllabus, avoid using it here**

1. Define a list that stores 5 numbers of your choice. Print them one by one using a for loop then do it using a while loop.
2. Define a list l with l = [1,2,3,4,5,6,7,8,9]. Use a while loop then a for loop to print every second element i.e. print 2,4,6,8
3. Define a tuple t to be t = (1,4,6,3) and a list l to be l = [‘0’,’1’,’2’,’3’,’4’,’5’,’6’,’7’,’8’]. Use a for loop to print the numbers in l given by the indices in t i.e. it should print 1,4,6,3 but the numbers should come from l.
4. Use a for loop to fill an empty list with 7 random floats between 0 and 1, print the list and print the average of the list. (Hint: Use random.random())
5. Use 2 for loops to generate the list [[1,2,3], [1,2,3], [1,2,3], [1,2,3]]
6. Define a function called list\_cap(l) that capitalizes all the elements of the list of strings l and returns the capitalized list. e.g.

>>> list\_cap([‘jerry’,’sipho’,’BRENDA’])

[‘Jerry’,’Sipho’,’Brenda’]

1. Define a function called is\_same(l1,l2) that returns True if l1 is the same is l2. (l1 and l2 can be any type!)
2. Define a function called has\_common(l1,l2) that returns True if l1 has at least 1 element in common with l2.

## 7.4: These lists be poppin’

1. Define a function called popL(l) that takes the elements in the list l, pops each element, prints it and prints what’s left in the list for each element in the list. e.g. if l = [1,2,3] it should print. (Hint: If you are using a for loop, you have to copy the list. You can also use a while loop.) Instructor should do both to explain why copying is necessary with a for loop.

>>> popL([1,2,3])

3

[1,2]

2

[1]

1

[]

1. Define a function called removeL(l1, l2) that removes all the items in l1 that are in l2 and returns a modified copy of. e.g. removeL([1,2,3,4,5],[2,3]) returns [1,4,5].
2. What is the difference between list.pop(i), list.remove(element) and del list[i]?
3. Why is it important to copy a list if you are going to iterate over it with a for loop and it is changing?
4. Find the errors in the following code blocks/snippets and debug them

#1

l = [1,2,3,4,5]

pop(3)

#2 – Why does this print None?

l = [1,2,4]

print(remove(2))

#3

l = [1,2,3]

print(l.pop(3))

#4

for item in myList:

print(item)

#5

l = [“1”,”hat”,”45”,”ayaya”]

for item in l:

l.remove(item)

#6

l = [1,3,4,6]

for item in l:

l\_copy = l[:]

print(l\_copy.pop(0)

#7

l = []

for i in range(67):

l = l.append(i)

#8

l = []

i = 0

while i < 15:

l.append(i)

# 8: Functions II

## 8.1: Global Warming

1. Define a variable called
2. Define a function called temp\_convert(T) that converts the Temperature T from Fahrenheit to Kelvin.

You are hired by a publishing company that publishes hundreds of books a year. They hire editors to read all the books and make sure there are no mistakes. It wastes a lot of time for the editors to check small things like punctuation so they ask you as a master programmer to develop software that tells them where there are mistakes with punctuation quickly then they fix them.

When writing, a period (full stop) should be followed by a space unless it is the end of the paragraph, you are going to write a function called find\_error(s) that takes in a paragraph as a string **s** and returns all the indices of where there is no space after a full stop.

e.g.

>>> find\_error(“Mistake not here. Mistake here.Mistake not here.”)

>>> [30]

#1 - student solution - show them this one

def find\_error(s):

"""

Returns the indices of the periods that are not followed

by spaces unless they are at the end

of the paragraph s

s -> str

returns list

"""

errors = [] #list

ind = 0

while ind < (len(s)-1):

char = s[ind]

if char == '.':

if ind < len(s)-1 and s[ind+1] != ' ':

errors.append(ind)

ind += 1

return errors

print(find\_error("Mistake not here. Mistake here.Mistake not here."))

#2 - tutor solution - liyakuhumbula pho lokhu? Asazi

def find\_error(s):

"""

Returns the indices of the periods that are not followed

by spaces unless they are at the end

of the paragraph s

s -> str

returns tuple

"""

errors = []

for ind,char in enumerate(s):

if char is '.':

if ind < len(s)-1 and s[ind+1] != ' ':

errors.append(ind)

return errors

print(find\_error("Mistake not here. Mistake here.Mistake not here."))