Name: Group: School:

**Quiz 2 [Out of 45]**

In this quiz, you are allowed to use any source of information except another person, you can use notes, the internet or any books, **except for part A.** If you ask for help from a tutor, they will help you but points might be deducted. Have fun!

**Part A: Short Questions**

1. In a few words, explain what the following things mean. No stories please [4].

**Accept any answer that makes sense, don’t have to be exact. [1] for each**

* 1. Calling a function

Making a defined function do something

* 1. Looping

Doing something over and over

* 1. Flow Control

Decision making, Controlling which code runs

* 1. Data Structure

Object that stores other objects

1. Fill in the parameters to the following range function calls to produce the numbers shown. The first one has been done for you. [4]

**The stop parameter can take many values, as long as it works its fine. [1] for each**

|  |  |
| --- | --- |
| range(1,9,2) | 1,3,5,7 |
| range( 1,7,1 or 1,7 ) | 1,2,3,4,5,6 |
| range( 50,62,2 ) | 50,52,54,56,58,60 |
| range( 100,800,100 ) | 100,200,300,400,500,600,700 |
| range( 12,0,-2 ) | 12,10,8,6,4,2 |

1. Find the error in the following code blocks and explain what causes it. There is only one syntax error in each problem. [4]

**[1] for each**

No colon for function

def my\_func(x,y)

’’’

does nothing

’’’

return None

for i in range(14):

print(item)

x not defined

item not defined

while x < 4:

name = input(“What is your name?” )

Concatenating string and int

for item in [1,2,3,4,5]:

print(“Number “ + item)

**Part B: Short Programs**

You are allowed to use a computer or your notes for this part of the quiz. Do not ask your friends for answers! Only the answers on this paper will be marked.

1. **Data Structures [7]**

Write a script that does the following in the box below. Only answers written here will be marked.

**[1] for each line**

Define a list called **l** with the numbers 1,2,3

Prints the type of **l** (Hint: This is a script so you should actually use print())

Define a tuple called **t** with the numbers 4,5

Prints the type of **t** (Hint: This is a script so you should actually use print())

Define a string called **s** with the number 6

Prints the type of **s** (Hint: This is a script so you should actually use print())

Put them all together in a string and print it(Hint: What do you have to do to be able to join them together?)

l = [1,2,3]

print(type(l))

t = (4,5)

print(type(t))

s = ‘6‘

print(type(s)

print(l,t,s) or print(str(l)+str(t)+s)

1. **Control [8]**

Do **any 2** of the following questions.

* 1. Write a short program that draws an equilateral triangle of side 200 pixels using the turtle module (Hint: An equilateral triangle has interior angle 60 degrees, what complementary angle does the turtle have to turn?). Use any method you want, with or without loops.

**There are many ways to do these, if you are not sure check in a script! Don’t just assume it won’t work.**

import turtle [1] for importing

turtle.forward(200) [1] using forward() or fd()

turtle.left(120) [1] using left() or right()

turtle.forward(200)

turtle.left(120)

turtle.forward(200)

turtle.left(120) [1] for repeating 3 times

or

import turtle [1] importing

for i in range(3): [1] Using for loop properly

turtle.forward(200) [1] using forward()/fd()

turtle.left(120) [1] using left()/right()

* 1. Use a for loop to print all the elements in the tuple t = (1,45,65,76,23,75,23). You should define t in your code

t = (1,45,65,76,23,75,23) [1]

for i in t: [2]

print(i) [1]

* 1. Use a while loop to print all the elements in the tuple t = (1,45,65,76,23,75,23). You should define t in your code.

t = (1,45,65,76,23,75,23)

i = 0 [1]

while i < len(t)-1: [1]

print(t[i]) [1]

i += 1 [1]

1. **Defining functions [8]**

Do **any 2** of the following questions in the spaces given after the example.

1. Define a function called divisible\_by\_13(n) which returns True if n is divisible by 13 and False otherwise.

>>>divisible\_by\_13(13)

True

>>>divisible\_by\_13(12)

False

def divisible\_by\_13(n): [1]

if n%13 == 0: [1]

return True [1]

else:

return False [1]

or

def divisible\_by\_13(n): [1]

if n%13 == 0: [1]

return True [1]

return False [1]

**Some people might put a different condition that works. Check! Lingalahlisi abantu mahara**

1. Define a function called quad(a,b,c) that returns the solution with a positive sign to a quadratic equation given by f(x) = a\*x\*\*2 + b\*x +c = 0. Remember the quadratic formula from O level (You should know it by heart!) is given by:

>>> quad(1,2,1)

-1

>>> quad(1,4,1)

-0.2679491924311228

Imagine if you had this during your exams!

1. Define a function called append\_two(l, x, y) which adds x and then y to the end of the list l.

>>> append\_two([1,2,3],4,5)

[1,2,3,4,5]

>>> append\_two([‘a’,’b’],’c’,’d’)

[‘a’,’b’,’c’,’d’]

def append\_two(l,x,y): [1]

l.append(x) [1]

l.append(y) [1] for using append y

return l [1]

**or**

**The best solution is the one below but you probably didn’t teach them how to copy lists so the one above is fine**

def append\_two(l,x,y): [1]

myl = l[:] [1] for copying

myl.append(x) [1]

myl.append(y)

return myl [1]

**Part C: Long Question [14]**

1. You are hired by the Zimbabwe Anticorruption Unit to audit the voter’s roll i.e. to check that the voter’s roll does not have errors that allow rigging. Your job is to check to make sure that no voter appears more than once. We will assume that all Zimbabweans have different names i.e. there cannot be 2 people called Tapiwa. The ZEC gives you a python list containing the names of all the people on the voters roll. You are going to define a function called **audit**(*voters\_roll*) that prints all the people who are “riggers” and how many times they appear in the voters\_roll list. Use the guideline below or any method you want to do this.

Your function should do the following when called:

>>> audit(["Ben","Tom","Tom"])

Tom appears 2 times

>>> audit(["Ben","Tom"]) #No repetitions so does nothing

>>> audit(["Ben","Tom","Tom","Tom"])

Tom appears 3 times

This is a guideline for how you can implement your code. Your code should be able to run after every step. Only continue if your code runs! If you cannot fix it, ask a tutor and you will lose one point.

* 1. Create a new script and name it with your first name, last name and school (Remember the rules for naming files) and save it in your folder (C:/users/zimcode/Desktop).

**If you cannot do part a. ask for help, you will not be penalized.**

* 1. On the first line of your script write a comment with your name and school.
  2. Begin your function definition as given below. The list called **riggers** is where will keep the names of all the riggers. You do not need to include the docstring in your code.

def audit(voters\_roll):

’’’

Checks the list voters\_roll to see if any elements appear

more than once

voters\_roll -> list

returns None

’’’

riggers = []

* 1. **FOR** each **voter** in the **voters\_roll** first check **IF** a voter appears more than 1 time using *listl.count(****voter****)* where list is replaced by the list containing the voters roll, if the **voter** appears more than 1 times **AND** the **voter** is **NOT** already **IN** the **riggers** list (to avoid duplicates), add them to the end of the **riggers** list using *list.append(****voter****).* Watch your indentation! This can be 3-6 lines of code.
  2. **FOR** each **rigger** in the **riggers** list, **PRINT** each rigger’s name and how many times they appear in the **voters\_roll** as shown in the examples above.

#Name School

def audit(voters\_roll):

riggers = []

for voter in voters\_roll:

if voters\_roll.count(voter)>1 and voter not in riggers:

riggers.append(voter)

for rigger in riggers:

print(rigger, "appears" ,voters\_roll.count(rigger), "times")

[1] for comment with name

[1] for definition

[1] for riggers definition

[1] for using loop (for or while) on voters\_roll

[1] for using if statement

[1] for voters\_roll.count >1 condition or something else that works

[1] for voter not in riggers condition

[1] for either using the and key word or if they used two if statements, they must be properly nested (indented) one inside the other

[1] appending to riggers

[1] for /while loop on riggers

[1] for printing a rigger (doesn’t have toprint number of times rigger appears)

[1] for printing number of times each rigger appears and concatenating

[2] Mbasela for anyone who at least tried, bedlula ama tutor

**There are many ways to do some of these parts! If you are not sure, test it or send me a picture and I will tell you if it works!**