|  |
| --- |
| Logbook for ISD |
| **Antonios Dimitriadis 21360003** |
|  |
|  |
|  |

Logbook for ISD

Antonios Dimitriadis 21360003

# Contents

[Contents 1](#_Toc503660165)

[Introduction 3](#_Toc503660166)

[Week 1 4](#_Toc503660167)

[Exercises 1 4](#_Toc503660168)

[Exercises 2 4](#_Toc503660169)

[Exercises 3 4](#_Toc503660170)

[Exercises 4 4](#_Toc503660171)

[Week 2 4](#_Toc503660172)

[Exercises 1 4](#_Toc503660173)

[Exercises 2 5](#_Toc503660174)

[Exercises 3 5](#_Toc503660175)

[Exercises 4 6](#_Toc503660176)

[Exercises 5 6](#_Toc503660177)

[Week 3 7](#_Toc503660178)

[Exercises 1 7](#_Toc503660179)

[Exercises 2 7](#_Toc503660180)

[Exercises 2a 7](#_Toc503660181)

[Exercises 2b 8](#_Toc503660182)

[Exercises 3 8](#_Toc503660183)

[Exercises 4 9](#_Toc503660184)

[Week 4 9](#_Toc503660185)

[Exercises 1 9](#_Toc503660186)

[Exercises 2 9](#_Toc503660187)

[Exercises 3 10](#_Toc503660188)

[Exercises 4 10](#_Toc503660189)

[Exercises 5 10](#_Toc503660190)

[Exercises 6 11](#_Toc503660191)

[Exercises 7 11](#_Toc503660192)

[Exercises 8 12](#_Toc503660193)

[Exercises 9 12](#_Toc503660194)

[Exercises 10 13](#_Toc503660195)

[Week 5 14](#_Toc503660196)

[Exercises 1 14](#_Toc503660197)

[Exercises 2 15](#_Toc503660198)

[Exercises 3 16](#_Toc503660199)

[Exercises 4 16](#_Toc503660200)

[Exercises 5 17](#_Toc503660201)

[Exercises 6 17](#_Toc503660202)

[Week 6 18](#_Toc503660203)

[Exercises 1 18](#_Toc503660204)

[Exercises 2 19](#_Toc503660205)

[Exercises 3 19](#_Toc503660206)

[Exercises 3a 19](#_Toc503660207)

[Exercises 3b 20](#_Toc503660208)

[Exercises 3c 20](#_Toc503660209)

[Exercises 3d 20](#_Toc503660210)

[Exercises 4 21](#_Toc503660211)

[Exercises 5 22](#_Toc503660212)

[Exercises 7 22](#_Toc503660213)

[Exercises 8 23](#_Toc503660214)

[Week 7 23](#_Toc503660215)

[Exercises 1 23](#_Toc503660216)

[Exercises 2 25](#_Toc503660217)

[Exercises 3 25](#_Toc503660218)

[Exercises 4 26](#_Toc503660219)

[Exercises 5 26](#_Toc503660220)

[Week 8 27](#_Toc503660221)

[Exercises 1 27](#_Toc503660222)

[Exercises 2 27](#_Toc503660223)

[Exercises 3 28](#_Toc503660224)

[Exercises 4 28](#_Toc503660225)

[Exercises 5 29](#_Toc503660226)

[Exercises 6 29](#_Toc503660227)

[Exercises 7 30](#_Toc503660228)

[Exercises 8 30](#_Toc503660229)

[Exercises 9 30](#_Toc503660230)

[Week 10 31](#_Toc503660231)

[Exercises 1a 31](#_Toc503660232)

[Exercises 1b 31](#_Toc503660233)

[Exercises 2a 31](#_Toc503660234)

[Exercises 2b 32](#_Toc503660235)

# Introduction

The module “Introduction to software development” is divided in 10 units of theory and lab exercises. The module begins with brief introduction to hardware and software and look on programming. It gives you the indicative content of the course. The programming tools Code editor and Code repository and the programming language that will be used is Python.

Then the following unit, introduce you the definition of the program and the algorithm. After that it shows you how to translate problems to programs and pseudocodes. In addition, it presents an overview of the programming languages from assembly language to high level languages and it gives you historical description of the programming language. Python is the established language of this module which is a high-level programming language. Python read-eval-print loop language shell is Idle, it can be used only one command line at a time in the shell and the shell is an interpreter.

The following units give you a taste of python fundamentals of programming and developing. We learned how to run programs on idle how to create a file .py and run multiple command. Also, it gives us a lot of exercises for datatypes, operations and errors.

Moreover, for the flow control structures, data types, methods, functions and object-oriented programming concepts we done a lot of practice and theory to understand and learn all of that.

In the beginning of the module I didn’t have any prior experience to programming language and concepts. Due to the fact, that we were introduced in a lot of different concepts and information that were unknown in the past, sometimes I found a little bit demanding to keep up. But, surprisingly by the end of the module I was able to write and execute code in python and understand concepts of programming.

# Week 1

## Exercises 1

What is a code repository (often also called version control system) used for?

## Exercises 2

Why is it advantageous to use a code repository?

## Exercises 3

Describe the different “layers” of Software that exist on a typical computer and explain why there are different layers of software.

## Exercises 4

Describe what an algorithm is and explain why it is a useful “tool” to translate from a human level problem (we can think of) to a computer program.

In mathematics and computer science, an algorithm is a self-contained unambiguous sequence of well-defined actions to be performed.

An algorithm is an effective method that can be performed within a finite amount of space and time and in a well-defined formal language for calculating a function (solving problem). In other words, algorithms are a step by step solution to a problem.

Starting from an initial state and initial input the instruction describe a computation that, when executed, proceeds through a finite number of well-defined successive states, eventually producing an output and terminating at a final ending state.

The algorithm is needed to be translated from human level problem to computer language because a computer requires a formal language designed in order to communicate.

The algorithm is set of instructions that is executed step by step and tells a computer what to do in order to come up with a solution to a particular problem. Computer and programming require algorithm in order to work because computers are dump. Algorithm has well defined steps.

# Week 2

The issue of this session is the importance of algorithms in computer science and how helpful is for programming also the pseudocodes. We breakdown the Python program compiler, interpreter for programming in Python. The difference between statements and expressions, and we learn our first function input() you will see how we use that function in the exercises.

## Exercises 1

Write an algorithm that describes how to make scrambled eggs, try to use control words, like IF, WHEN, UNTIL, WHILE, WAIT, AND, OR.

1. Take the eggs from the fridge
2. Put the pan over a medium fire AND add butter
3. Beat the eggs UNTIL has just one consistency
4. IF the butter is melt add the eggs
5. OR pour the eggs and let it sit,
6. WHILE waiting, get a plate
7. WAIT for 1 minutes
8. Lifted and folded it over from the bottom of the pan.
9. WHEN the eggs are scrambled and one consistency you can remove and put in your plate.

## Exercises 2

Is Idle (the Python language shell) an Interpreter or a Compiler or both? Explain your answer.

First of all, I would like to discuss about the definition of Interpreter and Compiler.

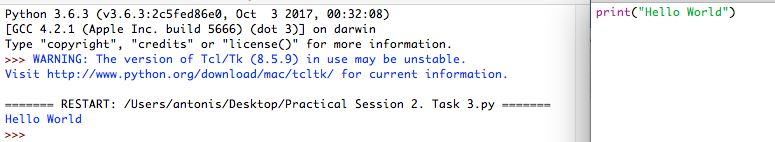
Compiled languages are those languages where source code is converted into native/machine one and that native piece of code is executed at run time and software/program unit we use for such languages are called compiler.

Interpreted languages are those one where source code is executed line by line at run time and software/program unit we use for such languages are called interpreter.

Idle (the Python language shell) is an interpreter we can run only one command at the time but that doesn’t mean python can run more commands. We can easily create a file .py and after we can have editor to write our command and compiled them. Python have both but especially python shell an interpreter.

## Exercises 3

Write a command in the Idle shell that says “Hello world”



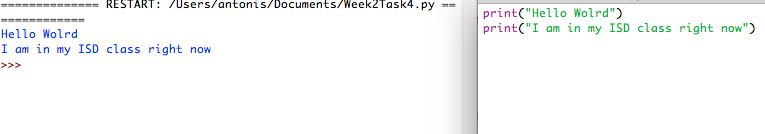
As you can see I type my command and executed in my Python shell (Idle). I create a new file .py(format) that is not necessary if you have only one line of command because shell is interpreter and you can run only one command at a time.

## Exercises 4

Write a program that produces the following output:

Hello World

I am in my ISD class right now



For this output is necessary to create file .py(format) because we have multiple command and as I mention before Idle can’t run this module because is interpreter Is allowing to run one command. This why we have to create and open new editor that allows multiple command that are being compiled and executed.

## Exercises 5

Write a program that asks the user for his/her name and produces an output like:

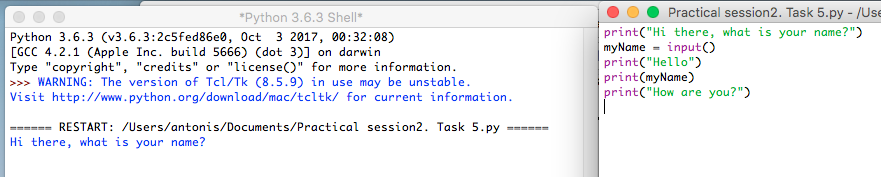
Hi there, what is your name?

>User input to be read<

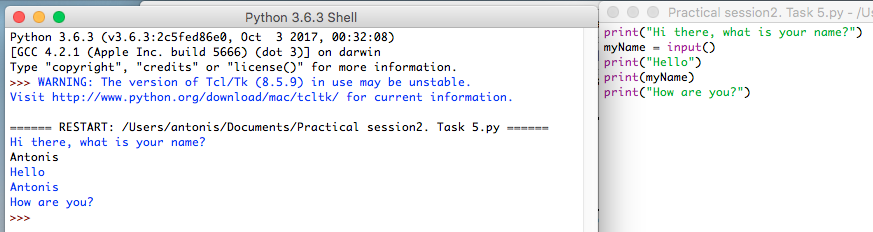
Hello

“User name”

How are you?



Here is printing only the first line and we have to give the name to continue the program.



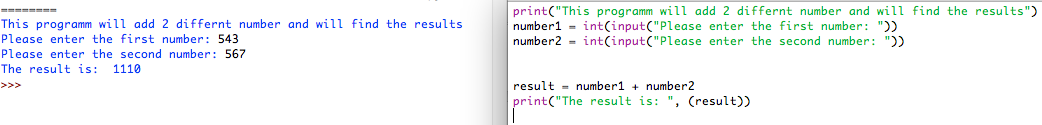
And as I gave my name is continue the program like a normal. That is happening from the function I use input().

# Week 3

In brief, we will see about the symbols in python what is the variables and objects. The datatypes of python and the converting types. There are following more exercises to understand and practice in python.

## Exercises 1

Write a program that asks for two numbers (Python has all the basic mathematical functions in place, like +,- etc.), adds them up and displays the result.



## Exercises 2

Answer the questions by implementing the code and run it.

## Exercises 2a

1. What will the output be from the following code?

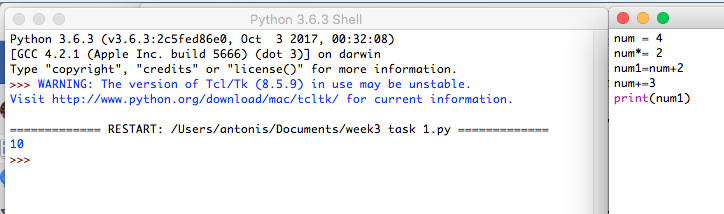
num = 4

num\*=2

num1=num+2

num1+=3

print(num1)



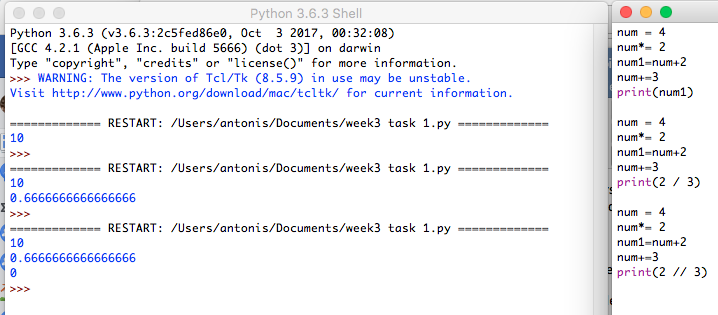
The output for this expression is 10. Represents a result from evaluation.

## Exercises 2b

1. What do the following lines of code output? Why do they give a different answer?

print(2 / 3)

print(2 // 3)



The first is divide the left hand operand by right hand operand. Always results in a float.

The second is floor division that give result in whole number adapted to the left number. (Is rounded to the closest integer. We call this operator integer division as well.)

## Exercises 3

All of the variable names below can be used. But which of these is the better variable name to use?

A a Area AREA area areaOfRectangle AreaOfRectangle

Better variable name is areaOfRectanagle.

## Exercises 4

Which of these variables names are not allowed in Python? (More than one might be wrong.)

apple APPLE Apple2 1Apple account number account\_number account.number accountNumber fred Fred return return\_value 5Return GreatBigVariable greatBigVariable great\_big\_variable great.big.variable

Are not allowed in Python: 1Apple, account.number, account number, account#, great.big.variable

# Week 4

In this unit, will refer to the Python keywords, name conventions and how to use in a right way the data as well as converting types and operators whose function and use is important. In practice, we will use integer as well as float inputs, import functions and some errors, where there are logic errors, run time, syntax and common.

## Exercises 1

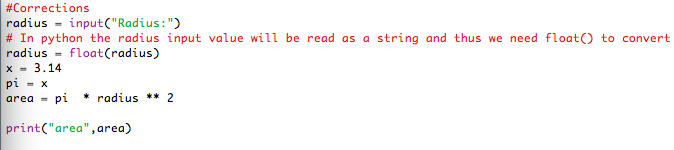
Explain the mistake in the following code:

radius = input("Radius:")

x = 3.14

pi = x

area = pi \* radius \*\* 2



C:\Users\21360003\Downloads\Week4\Week4Task1a.fw.png

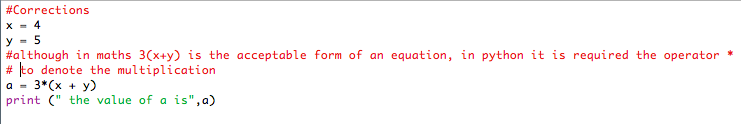
## Exercises 2

Explain the mistake in the following code:

x = 4

y = 5

a = 3(x + y)

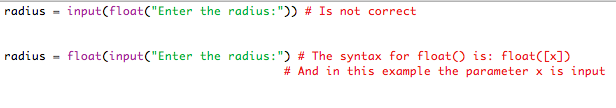


C:\Users\21360003\Downloads\Week4\Week4Task2a.fw.png

## Exercises 3

Explain the mistake in the following code:

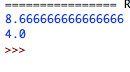
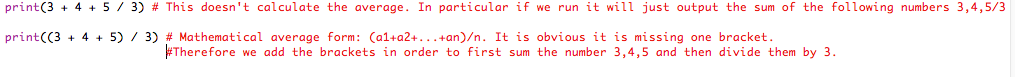
radius = input(float("Enter the radius:"))



## Exercises 4

Why does this code not calculate the average?

print(3 + 4 + 5 / 3)



## Exercises 5

Consider the following code:

x = 19.93

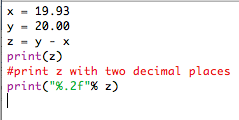
y = 20.00

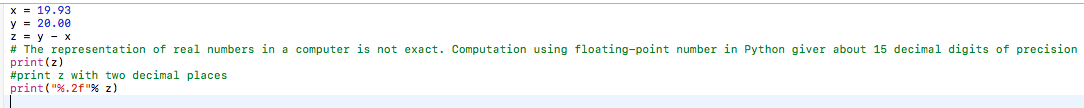
z = y – x

print(z)

The output is 0.0700000000028 Why is that so?

Improve the code so that the output is to two decimal places.

C:\Users\21360003\Downloads\Week4\Week4Task5a.fw.png



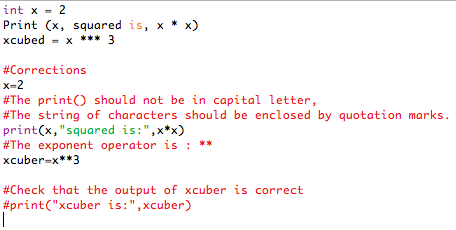
## Exercises 6

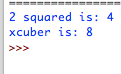
Find at least three compile-time errors:

int x = 2

Print (x, squared is, x \* x)

xcubed = x \*\*\* 3





## Exercises 7

Find two run-time errors:

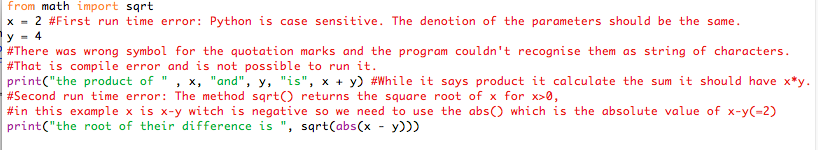
from math import sqrt

X = 2

Y = 4

print(“The product of “, x, “and”, y, “is”, x + y)

print(“The root of their difference is “, sqrt(x – y))



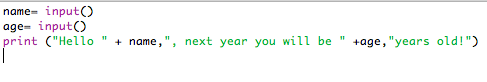
C:\Users\21360003\Downloads\Week4\Week4Task7.fw.png

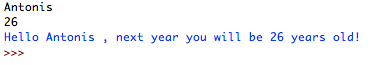
## Exercises 8

Write statements to prompt user for their name and age

Write a print statement to output:

Hello \_\_\_\_, next year you will be \_\_\_\_ years old!



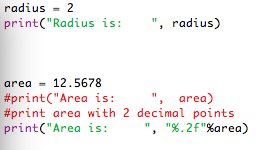


## Exercises 9

Given that radius is 2 and area is calculated as 12.5678, use string format operators to print the values of the variables radius and area so that the output looks like this:

Radius is: 2

Area is: 12.57

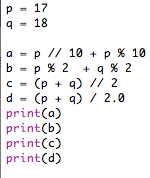


C:\Users\21360003\Downloads\Week4\Week4Task9a.fw.png

## Exercises 10

What are the values of the following expressions, assuming that p is 17 and q is 18?

1. p // 10 + p % 10
2. p % 2 + q % 2
3. (p + q) // 2
4. (p + q) / 2.0



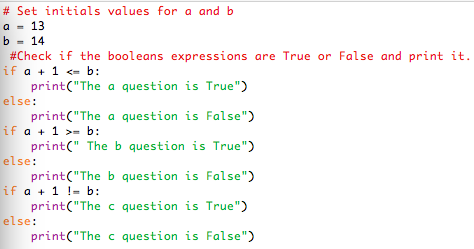


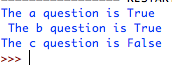
# Week 5

This five week course focus on flow control structures. In particular, we learned what is a variable and to analyse it, data types (integer, float, strings) and how to characterize a parameter in a specific type. Then we saw the python operators and the order of operations and parentheses. Finally, we learned how to escape sequences with strings and formatted output specifiers (e.g how to print a number with 2 digits after the decimal place).

## Exercises 1

1. Which of the following conditions are true, if **a = 13** and **b = 14** ?
2. a + 1 <= b
3. a + 1 >= b
4. a + 1 != b



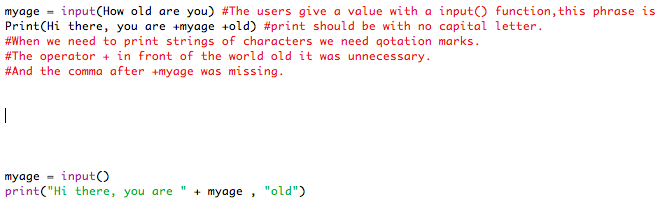


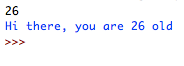
## Exercises 2

Explain the mistake(s) in the following code:

myage = input(How old are you)

Print(Hi there, you are +myage +old)





## Exercises 3

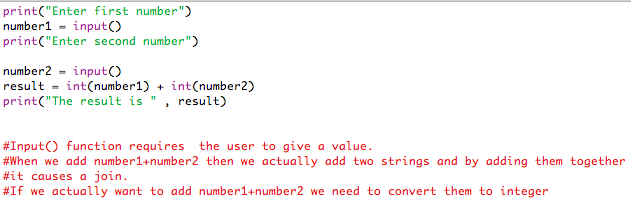
Explain why the following code won’t really add the two “numbers”:

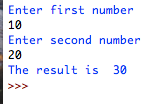
number1 = input("Enter first number")

number2 = input("Enter second number")

result = number1 + number2

print(“The result is ”+result)





## Exercises 4

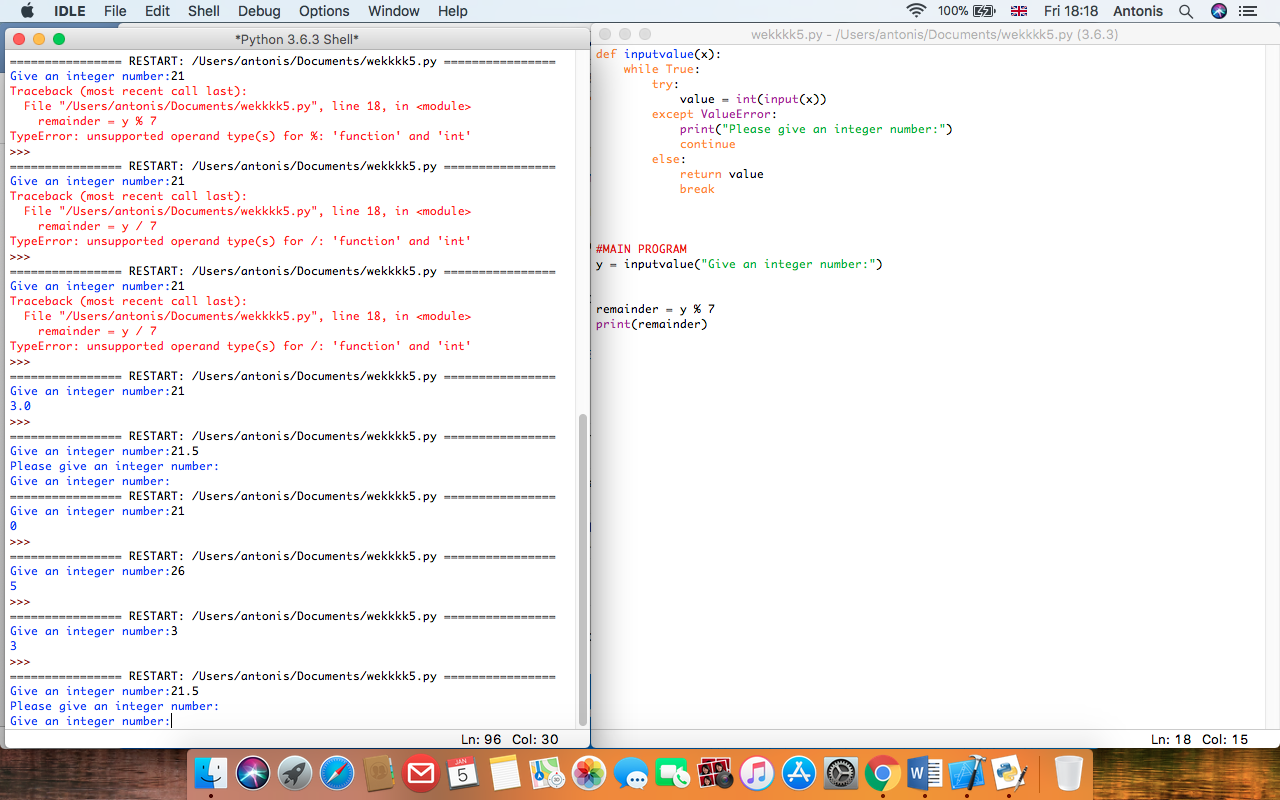
Write code to calculate the average of: {3, 11, 78, 112, 4, 18} in one single line of code.

C:\Users\21360003\Downloads\week5\Week5Task4.fw.png

C:\Users\21360003\Downloads\week5\Week5Task4a.fw.png

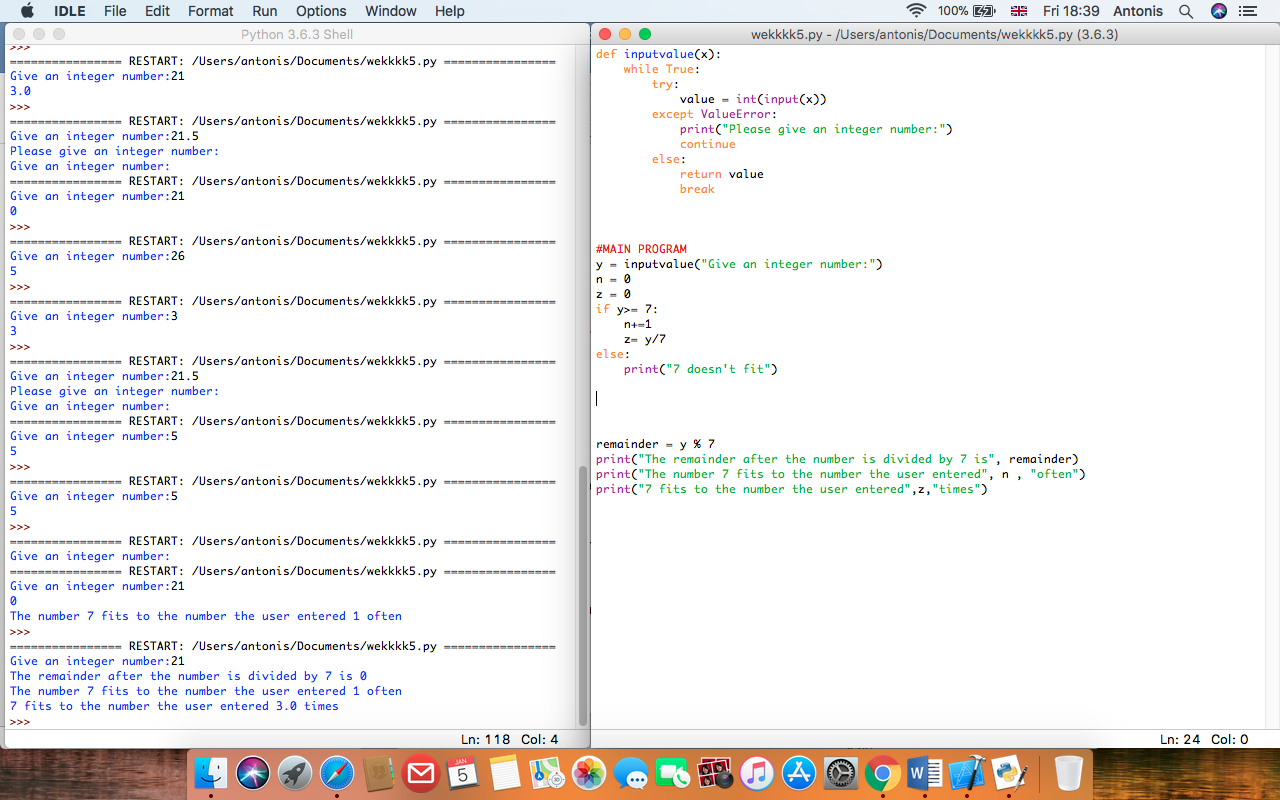
## Exercises 5

Write a program that asks the user for an integer number and then prints out the **remainder** after the number is divided by 7.



## Exercises 6

Expand the above program (5.) by also printing out how often the number 7 “fits” into the number the user entered.



# Week 6

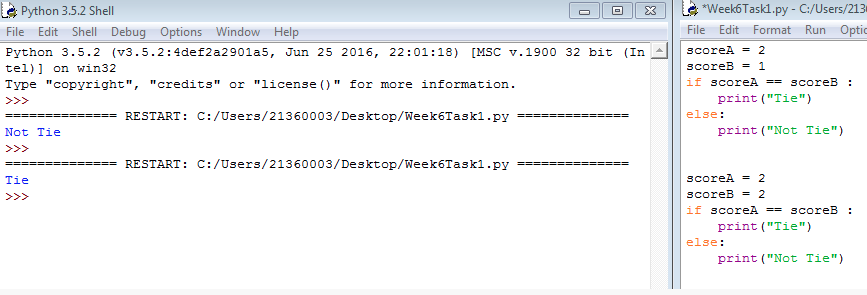
The following session present flow control structures. We learned how to check a condition with if statement and repeat a set of statements with the “while” and “for” statement. We saw a lot of examples with if statement (one way, multiple, nested). Another part of this week was the logical operators AND, OR, NOT and the Boolean truth tables.

## Exercises 1

What is the error in this statement?

if scoreA = scoreB :

print("Tie")



The values should be compared with ==, not =.

## Exercises 2

Supply a condition in this if statement to test if the user entered a “Y”:

userInput = input("Enter Y to quit.")

if . . . // supply statement

print("Goodbye") // if the user entered “Y”



## Exercises 3

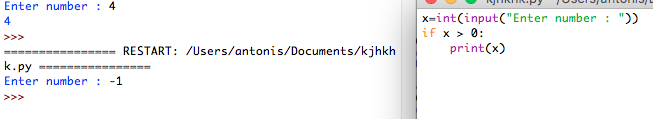
Find the errors in the following if statements, correct where necessary.

## Exercises 3a

if x > 0 then :

print(x)

Screen%20Shot%202018-01-13%20at%2020.45.57.png



The program is running without any error. As then was the error in the command

## Exercises 3b

if 1 + x > x \*\* sqrt(2) :

y = y + x

1. Correction: import math is missing
2. Error: 1+
3. Correction: math.sqrt(2)

## Exercises 3c

if x = 1 :

y = 1

Correction:

if x == 1 :

y += 1

## Exercises 3d

letterGrade = "F"

if grade >= 90 :

letterGrade = "A"

if grade >= 80 :

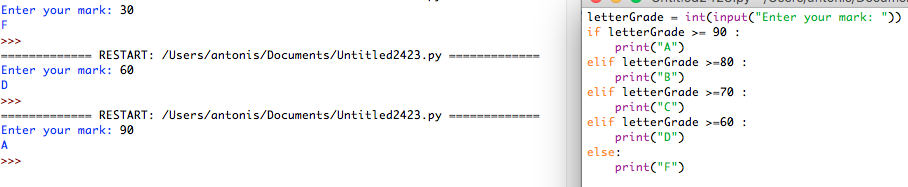
letterGrade = "B"

if grade >= 70 :

letterGrade = "C"

if grade >= 60 :

letterGrade = "D"



The elif is short for else if. It allows us to check for multiple expressions.

If the condition for if is False, it checks the condition of the next elif block and so on.

If all the condition are False, body of else is executed.

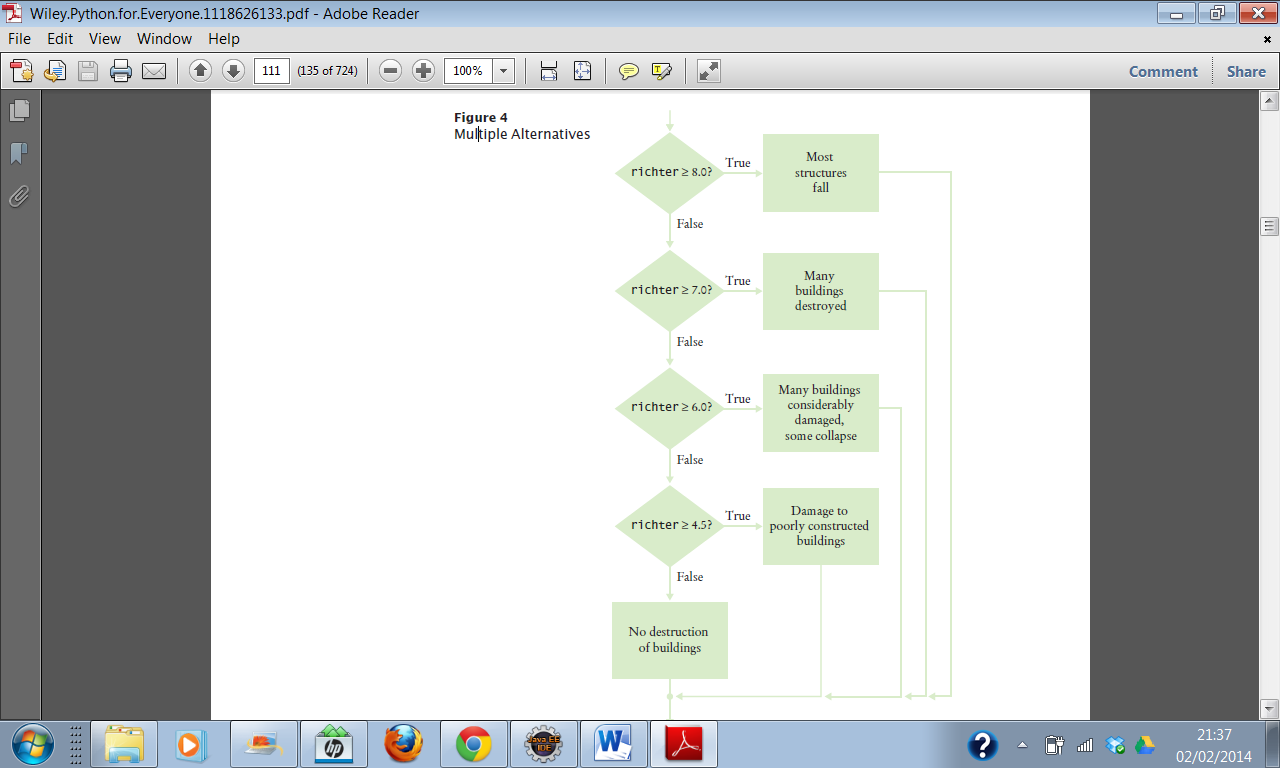
Only the one block among the several if…elif…else blocks is executed according to the condition. Whereas on the initial program it tests all the if statements whitch can lead on an error. For example if grade>=90, first it will assign the A to the lettergrade, but after it will test the other if conditions and it will be true for each of them because if grade>=90 then it is also greater than 80,70,60,50 e.t.c and thus the letter grade value will change multiple times till the the last if condition.

The if block can have only one else block. But it can have multiple elif blocks.

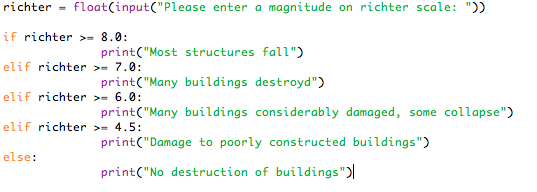
## Exercises 4

Using the flow chart below, construct the *if, elif, else* control structure necessary to implement the flow chart.

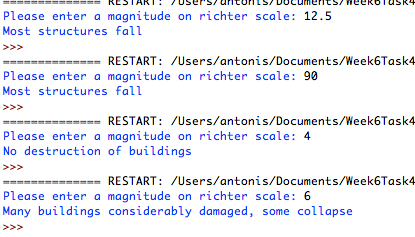
Complete your program to describe the earthquake by asking the user to enter a magnitude on the Richter scale and print out the effect that magnitude would have had (e.g. “Many buildings destroyed”).



The flow chart control should be like this:



We can run the program and get the correct answers:



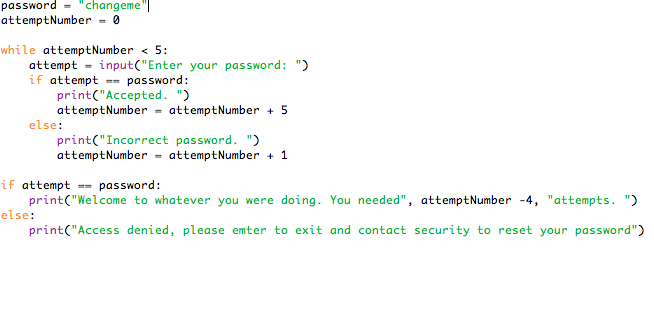
## Exercises 5

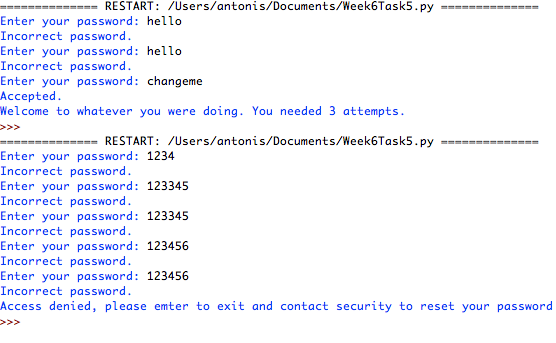
Write a program that sets a password as “changeme” and asks the user to enter the password and keeps asking until the correct password is entered and then says “Accepted”.

The program should count how many attempts the user has taken and tell them after they have been accepted.

*Extra Challenge:*

If the user takes more than 5 attempts the program should say, “Access denied, please press enter to exit and contact security to reset your password”





## Exercises 6

What do the following nested loops display? Hand trace.

for i in range(3) :

for j in range(1, 4) :

print(i + j, end="")

print()

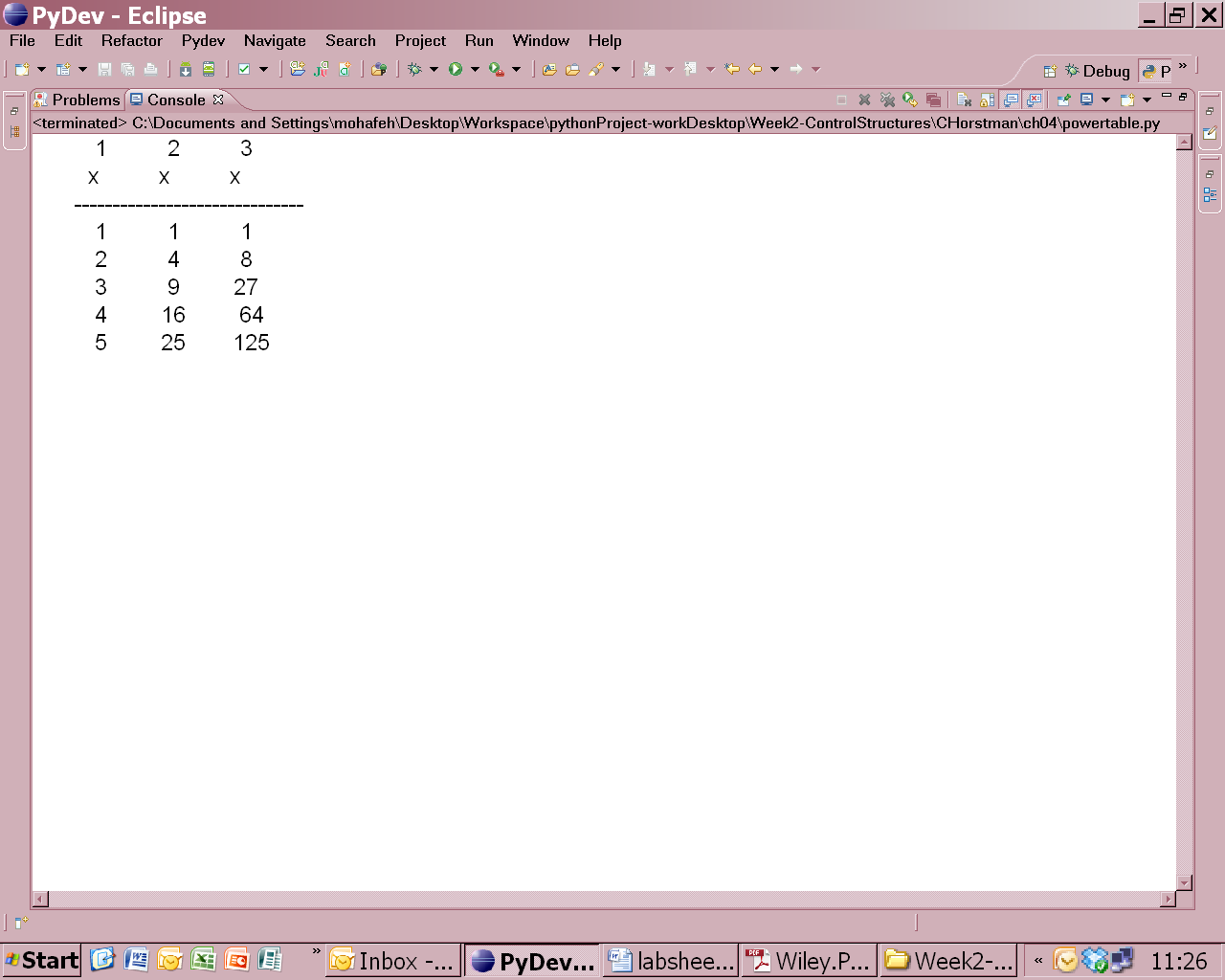
|  |  |  |  |
| --- | --- | --- | --- |
|  | i | j | Printed result |
| 1st iteration | 0 | 1 | 1 |
| 2nd iteration | 0 | 2 | 2 |
| 3rd iteration | 0 | 3 | 3 |
| 4th iteration | 1 | 1 | 2 |
| 5th iteration | 1 | 2 | 3 |
| 6th iteration | 1 | 3 | 4 |
| 7th iteration | 2 | 1 | 3 |
| 8th iteration | 2 | 2 | 4 |
| 9th iteration | 2 | 3 | 5 |

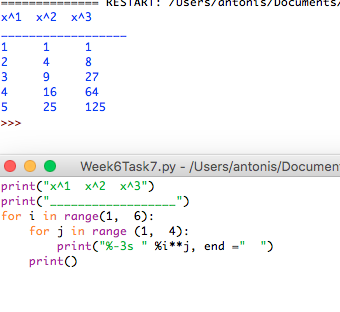
This program presents a nested loop. The I will take value from 0 to 2 and j from 1 to 4.

The result are illustrated above for each iteration.

## Exercises 7

Write a program that will generate a table to print powers of the first 5 numbers. Your output should be similar to the sample given below.





# Week 7

Explains the categories of Data structures which are string, lists, tuples and dictionaries.

It explains in detail each category and their differences also shows the syntax in Python.

## Exercises 1

Assume the following list: **a = [ 66.25, 333, 333, 1, 1234.5 ]**

If you perform the following operations on the list:

a.insert(2, -1)

a.append(333)

What will the list look like?

Now you perform:

a.index(333)

What will the output of this operation be?

Now you perform:

a.remove(333)

What will the list look like?

Now you perform:

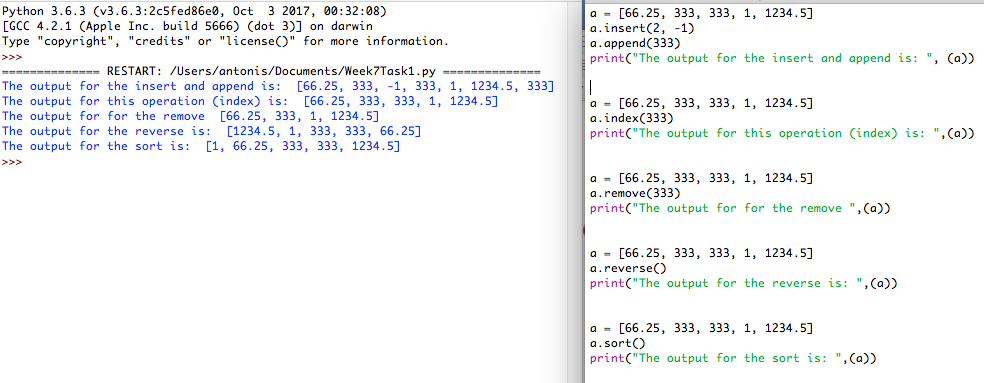
a.reverse()

What will the list look like?

Now you perform:

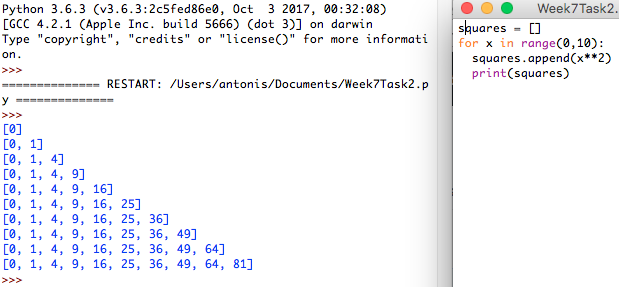
a.sort()

What will the list look like?



## Exercises 2

Write a short program to create a list of squares for numbers up to 10.   
Start with an empty list called squares and append squares of numbers from 0 up to 10. Print the contents of your list.



## Exercises 3

nums = []

for x in [1,2,3]:

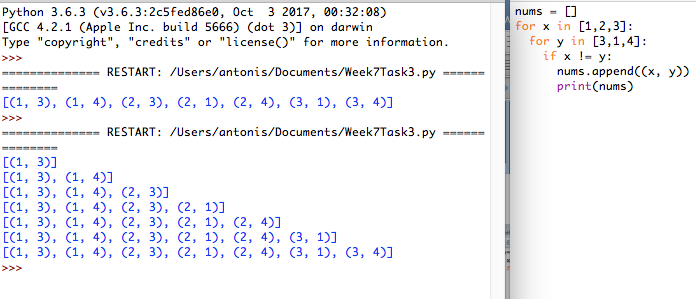
for y in [3,1,4]:

if x != y:

nums.append((x, y))

print(nums)

What is the output from above?

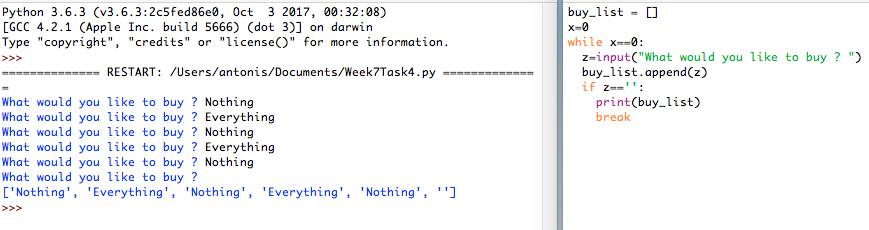


## Exercises 4

Create a program that will keep track of items for a shopping list.

The program should start with an empty list and keep asking for new items until nothing is entered (no input followed by enter/return key).

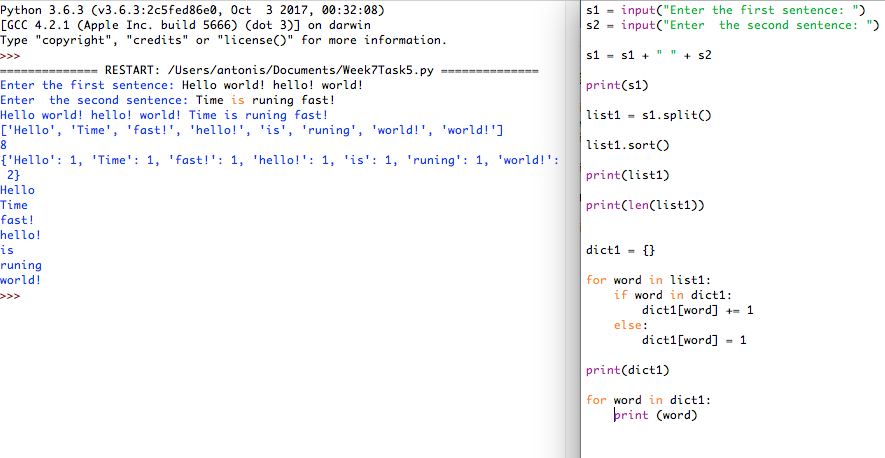
The program should then display the full shopping list.



## Exercises 5

Write a program that will ask the user to enter two short sentences and then:

* Concatenate the two sentences into one long sentence
* Split the sentence into a list of words
* Sort the words in alphabetical order and print them out
* Print the total number of words contained in your list
* Create a dictionary that will store each word together with the count of the occurrence of each word in your sentence.
* Print each item from the dictionary



# Week 8

In this week, it is defined what is a function, how to create and call a function and why is important to use blocks of functions in a program. Then it gives an extensive analyses of how the function is working along with a main program and it gives plenty of examples of some basic function.

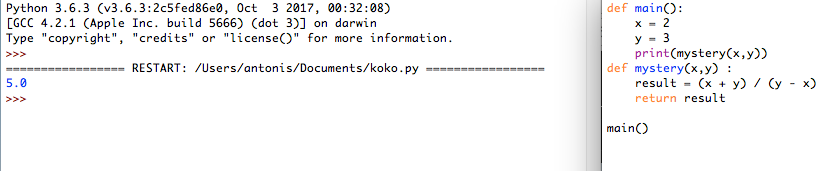
## Exercises 1

def mystery(x, y) :

result = (x + y) / (y - x)

return result

What is the result of the call mystery (2, 3)?



## Exercises 2

What does this program print?

def main () :

a = 5

b = 7

print(mystery(a, b))

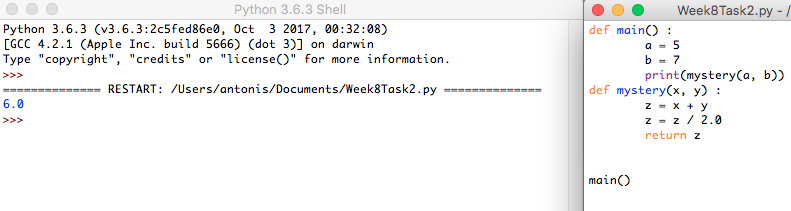
def mystery(x, y) :

z = x + y

z = z / 2.0

return z

main()



## Exercises 3

What does this program print?

def main() :

a = 4

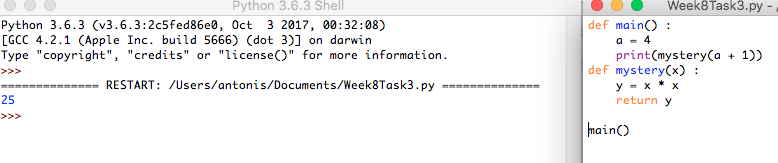
print(mystery(a + 1))

def mystery(x) :

y = x \* x

return y

main()



## Exercises 4

Consider this function that prints a page number on the left or right side of a page:

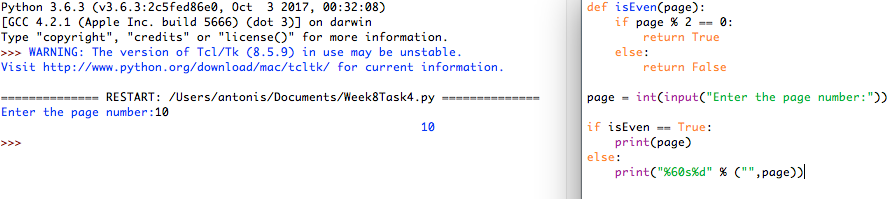
if page % 2 == 0:

print(page)

else :

print("%60s%d" % (" ", page))

Introduce a function that returns a Boolean to make the condition in the if statement easier to understand.



## Exercises 5

Transform the following instructions into a function called count\_spaces. Define a main function that will ask the user to enter some input and call the count\_spaces function to return the number of spaces.

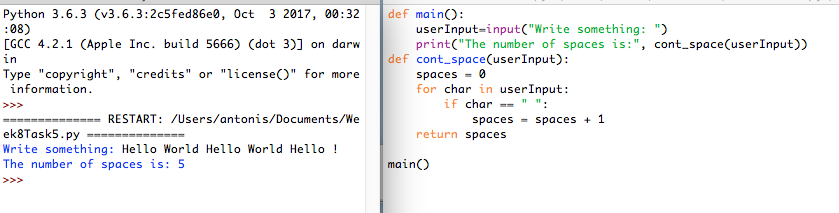
# Counts the number of spaces

spaces = 0

for char in userInput :

if char == " " :

spaces = spaces + 1



## Exercises 6

Consider this recursive function:

def mystery(n) :

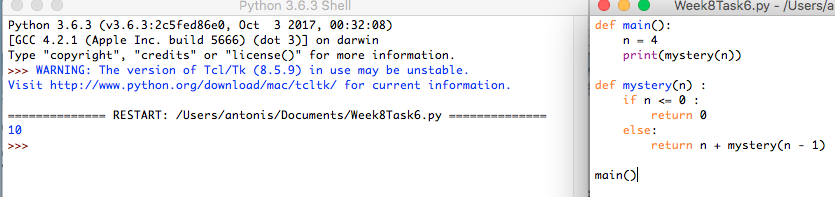
if n <= 0 :

return 0

else:

return n + mystery(n - 1)

What is mystery(4)?



## Exercises 7

def mystery(n) :

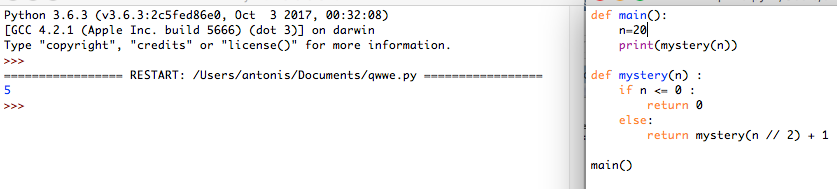
if n <= 0 :

return 0

else:

return mystery(n // 2) + 1

What is mystery(20)?



## Exercises 8

Consider these functions:

def f(x) :

return g(x) + math.sqrt(h(x))

def g(x) :

return 4 \* h(x)

def h(x) :

return x \* x + k(x) - 1

def k(x) :

return 2 \* (x + 1)

Without actually compiling and running a program, determine the results of the following function calls:

a. x1 = f(2)

b. x2 = g(h(2))

c. x3 = k(g(2) + h(2))

d. x4 = f(0) + f(1) + f(2)

e. x5 = f(-1) + g(-1) + h(-1) + k(-1)

## Exercises 9

Consider the following function:

def f(a) :

if a < 0 :

return -1

n = a

while n > 0 :

if n % 2 == 0 : # n is even

n = n // 2

elif n == 1 :

return 1

else :

n = 3 \* n + 1

return 0

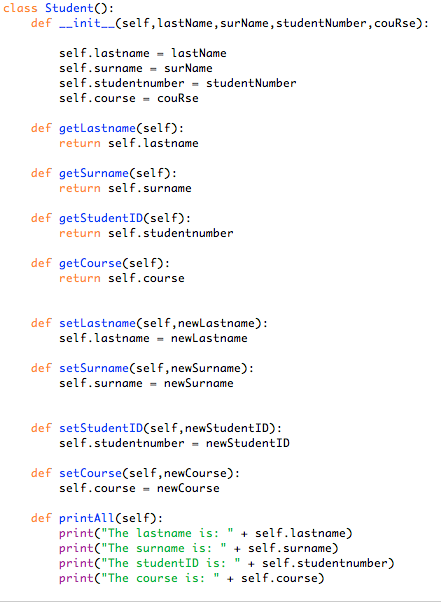
Perform traces of the computations f(-1), f(0), f(1), f(2), f(10), and f(100).

# Week 10

The aim of this unit is to present as the object-oriented programming concepts. It starts by explaining how to define and access the object and classes. Mostly focuses on to make us what a class is and what an object of a class is. Then it shows us a special method to construct an object “Constructor”.

## Exercises 1a

Write a class “Student” with a state that stores information about the student’s surname, last name, student number and course. The behaviour of the “Student” class should enable a student object to get and set each of the Student class variables and to print out all of the Student’s information.



## Exercises 1b

Write a program with a main function that imports the Student class, creates student object and prints and changes some of the information (like surname, last name) of the student object.



## Exercises 2a

**a)** A simulated cash register that tracks the item count and the total amount looks like this: due

class **CashRegister** :

def **\_\_init\_\_**(*self*) :

*self*.\_itemCount = 0

*self*.\_totalPrice = 0.0

def **addItem**(*self*, price) :

*self*.\_itemCount = *self*.\_itemCount + 1

*self*.\_totalPrice = *self*.\_totalPrice + price

def **getTotal**(*self*) :

return *self*.\_totalPrice

def **getCount**(*self*) :

return *self*.\_itemCount

def **clear**(*self*) :

*self*.\_itemCount = 0

*self*.\_totalPrice = 0.0

## Exercises 2b

Write a **TestRegister class** to test the **addItem**, **getTotal**, **getCount** and **giveChange** methods of the **CashRegister class**

