|  |
| --- |
| Logbook for ISD |
| Alexander Nicholas Zacharias 21323395 |
|  |
|  |
|  |

Logbook for ISD

Alexander Nicholas Zacharias 21323395

Contents

[Introduction 4](#_Toc469879082)

[Week 1 6](#_Toc469879083)

[Question 1 6](#_Toc469879084)

[Question 2 6](#_Toc469879085)

[Question 3 7](#_Toc469879086)

[Question 4 7](#_Toc469879087)

[Week 2 8](#_Toc469879088)

[Exercises 1 8](#_Toc469879089)

[Exercises 2 9](#_Toc469879090)

[Exercise 3 9](#_Toc469879091)

[Exercise 4 10](#_Toc469879092)

[Exercise 5 12](#_Toc469879093)

[Session 3 13](#_Toc469879094)

[Exercises 1 14](#_Toc469879097)

[Exercises 2 16](#_Toc469879098)

[Exercises 3 17](#_Toc469879099)

[Exercise 4 18](#_Toc469879100)

[Session 4 21](#_Toc469879101)

[Exercises 1 21](#_Toc469879102)

[Exercise 2 22](#_Toc469879103)

[Exercises 3 23](#_Toc469879104)

[Exercises 4 24](#_Toc469879105)

[Exercise 5 24](#_Toc469879106)

[Exercise 6 25](#_Toc469879107)

[Exercise 7 26](#_Toc469879108)

[Exercise 8 27](#_Toc469879109)

[Exercise 9 27](#_Toc469879110)

[Exercise 10 28](#_Toc469879111)

[Session 5 29](#_Toc469879112)

[Exercise 1 29](#_Toc469879113)

[Exercise 2 30](#_Toc469879114)

[Exercise 3 30](#_Toc469879115)

[Exercise 4 32](#_Toc469879116)

[Exercise 5 34](#_Toc469879117)

[Exercise 7 36](#_Toc469879121)

[Exercise 8 36](#_Toc469879126)

[Session 6 37](#_Toc469879127)

[Exercise 1 37](#_Toc469879128)

[Exercise 2 39](#_Toc469879129)

[Exercise 3 39](#_Toc469879130)

[Exercise 4 40](#_Toc469879131)

[Exercise 5 40](#_Toc469879132)

[Session 7 and 8 42](#_Toc469879133)

[Exercise 1 42](#_Toc469879134)

[Exercise 2 42](#_Toc469879135)

[Exercise 3 43](#_Toc469879136)

[Exercise 4 44](#_Toc469879137)

[Exercise 5 45](#_Toc469879138)

[Exercise 6 46](#_Toc469879139)

[Exercise 7 47](#_Toc469879140)

[Exercise 8 48](#_Toc469879141)

[Exercise 9 49](#_Toc469879142)

[Session 9 52](#_Toc469879143)

[Exercise 1 52](#_Toc469879144)

[Exercise 2 52](#_Toc469879145)

[Exercise 3 52](#_Toc469879146)

[Session 10 57](#_Toc469879147)

[Exercise 1 57](#_Toc469879148)

[Exercise 2 58](#_Toc469879149)

# Introduction

Within the module of Introduction to software development I have learned how to program in python, how to use code repositories and how to create logbooks like this one which I believe has helped me a lot and will continue to do so. The module itself has been quite challenging but with constant practice I got the hang of it and now find it quite easy to program in python. Moreover, whenever I am having a hard time with a piece of code in python I refer to this logbook which I have crafted over the past semester which often helps me figure out the problem I am having. I am very satisfied with my work so far and I hope to continue doing this in the years to come.

# Week 1

In week 1 of the course “Introduction to software Development” we talked about Learning Outcomes and learned how to access our module’s content on Blackboard but also were given some additional teaching and helpful material such as the code repository Github, the python course online and our textbooks for this module. Furthermore were given some names of code repositories and how the assessment works but also went into what the content of the course is, uses for computers, hardware, software what a program is.

## Question 1

**What is a code repository used for?**

A code repository is a place that software developers can store, backup or share all of their source codes which can be shared with anyone and can also be edited or improved by them, they can upload documents related to the project they are working on and can have multiple versions of your code which you can compare and/or get feedback on it. In addition to all of this you can also check where bugs begun and if you make a mistake you can easily go to an older version of the code and try again.

**References:**

*Stack Overflow* (2008)   
Available at: <http://stackoverflow.com/questions/1408450/why-should-i-use-version-control>

(Accessed: 29 September 2016)

## Question 2

**Why is it advantageous to use a code repository?**

The advantages of using a code repository or a version control system are many but the best one in my opinion is that it simply makes your life and work easier. One of the ways it does this is by allowing you to share your code with anyone and letting you get feedback by people who potentially have more experience dealing with a certain problem you might be having or maybe they have a way to make your code cleaner or better. Another advantage a version control system has is that you can monitor any change or progression that your code has received and even check where bugs might have started by simply looking at an older version of your code and even compare them to see what you might have done wrong or right since you can maintain any number of versions of your code in the code repository.

Stack Overflow (2008)

Available at: <http://stackoverflow.com/questions/1408450/why-should-i-use-version-control>

(Accessed: 29 September 2016)

## Question 3

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

The different “layers” of software that exist on a typical computer are the System software which consists of the operating system, utility programs and software development tools and the second layer which is Application Software which are the programs most of us use every day such as Microsoft word, Google chrome or even computer games.

The difference between the two layers is that the “System software” is an integral part which uses a lol-level language that helps the computer function correctly and provides a platform for the application software to be executed on. On the other hand “Application software” is created for users which use the applications for the task they need to accomplish.

**References:**

*Differencebetween*

Available at: <http://www.differencebetween.net/technology/software-technology/difference-between-system-software-and-application-software/>

(Accessed: 20 September 2016)

## Question 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.**

An algorithm in mathematics and computer science terms is a formula which is a step-by-step set of operations to be performed which include tasks like calculations, data processing and automated reasoning tasks.

The reason why algorithms are such a useful “tool” to use is the same reason why it’s useful for cooks to use recipes. It is a basic outline of procedures that must be followed in order for a program to work correctly and as smooth as possible. Furthermore we use them to make smart Artificial Intelligence,

# Week 2

The lecturer today introduced us to some program languages but focused more on the programming language, python. We were shown how to print “Hello world” and how values and integers work but we also learned about some errors that may come up in our code.

## 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. Get 4 Eggs.
2. Get frying pan.
3. Get fork.
4. Get butter.
5. Get mixing bowl.
6. Get salt.
7. Get pepper.
8. Get vegetable oil.
9. Get tablespoon.
10. Add 2 tablespoons of oil in pan.
11. Heat frying pan.
12. Wait UNTIL frying pan reaches 200 degrees.
13. WHILE frying pan is heating crack eggs into mixing bowl.
14. Add a pinch of salt.
15. Add a pinch of pepper
16. Beat eggs with fork.
17. WHEN frying pan at 200 degrees let it cool down to 100 degrees.
18. IF frying pan 100 degrees add beaten eggs.
19. WHEN you add beaten eggs stir slowly.
20. WHEN your eggs look silky and slightly runny they are ready.

## Exercises 2

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

The Python language shell, Idle is an interpreter for the reason that when Python uses the compile function the .pyc files get compiled into a python-specific code, called byte code. The byte code is not a machine-code and is instead executed by the Python Virtual Machine (PVM) which is the fundamental reason that the code is executed.

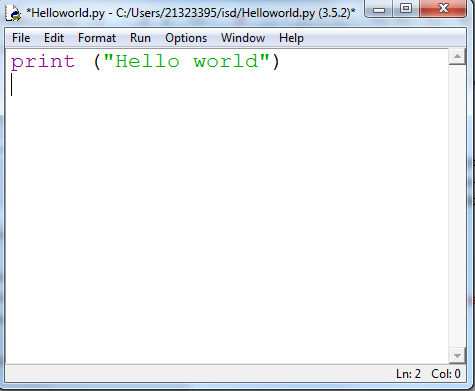
Reference:

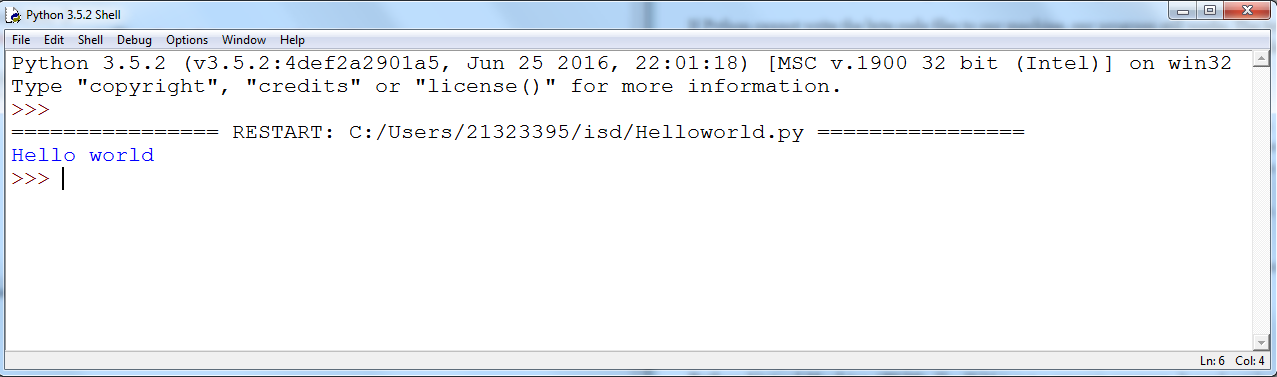
*Quora*

Available at: <https://www.quora.com/Is-Python-compiled-or-interpreted-or-both>

## Exercise 3

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



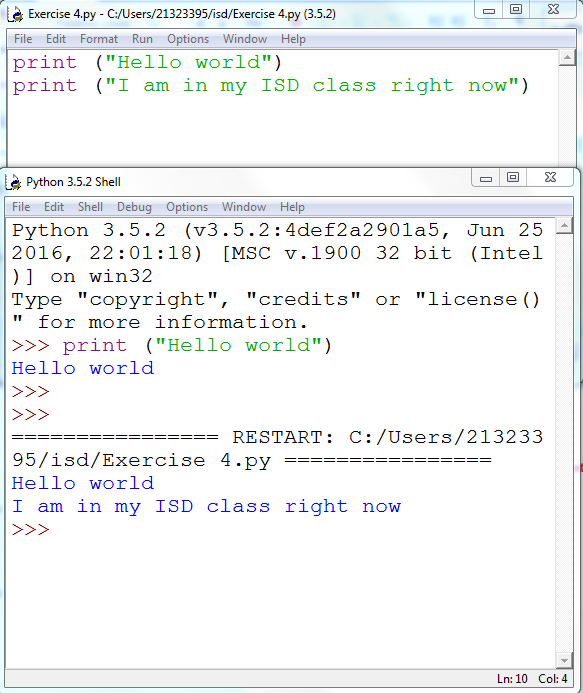


## Exercise 4

**Write a program that produces the following output:**

**Hello World**

**I am in my ISD class right now**



## Exercise 5

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

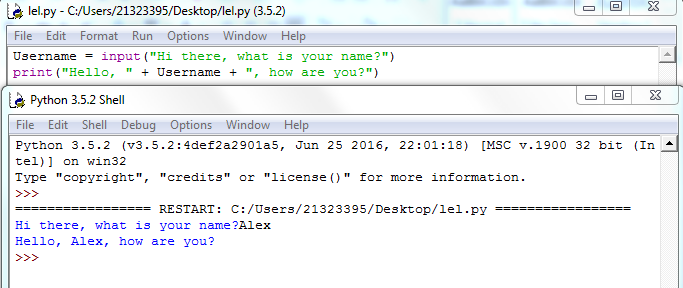
Hi there, what is your name?

>User input to be read<

Hello

“User name”

How are you?



# Session 3

In this week’s session we discussed how to use Github and how to commit our code. Moreover, we talked about the definitions of version Control and the code repository, why we should use the code repository and lastly we talked about the fundamentals of Python programming.

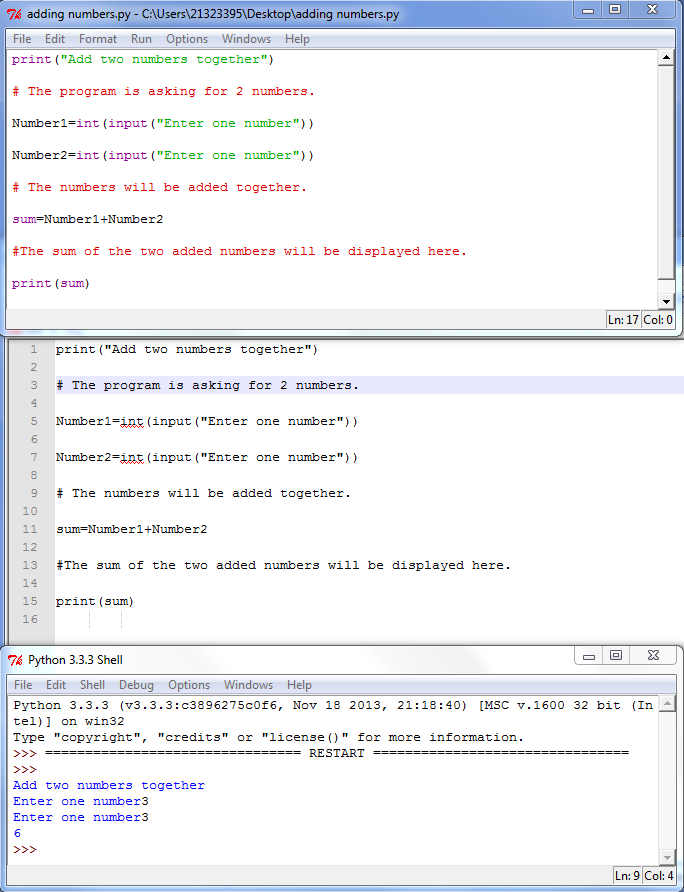
## Code repository: **Commit (upload) the code from week 1 to your repository.**

## 

The above image shows that I have created a Github account and uploading one of my codes that I had done in a practical session.

## 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.**

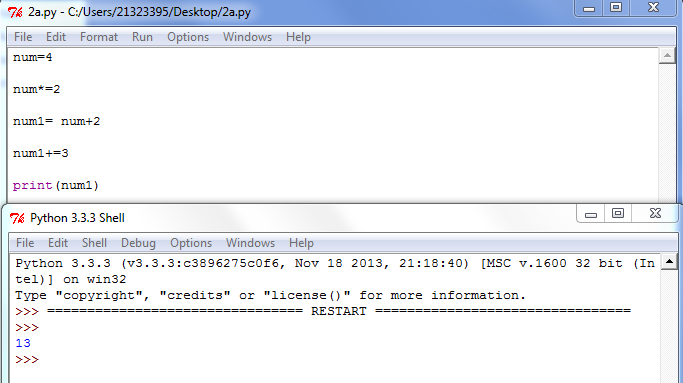


This program asks for 2 numbers and adds them together. I had an “invalid syntax” error so I used notepad++ to point out that I made a mistake on line 5 and 7 where I forgot to put the second parenthesis at the end on both lines.

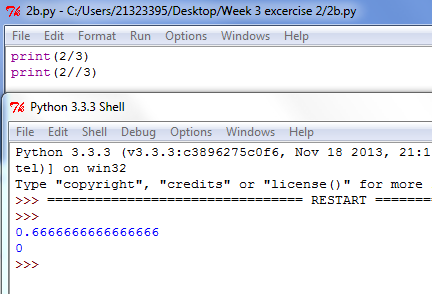
## Exercises 2

**Answer the questions by implementing the code and run it.**

2a)



2b)



The command “print(2/3)” preforms and action that divides the number 3 by 2 and gives us the result in the full form or float form ( 0.6666666666666666) but the command “print(2//3)” even though it preforms the same action which divides 3 by 2 it sends us the result in shortened form or an integer.

## Exercises 3

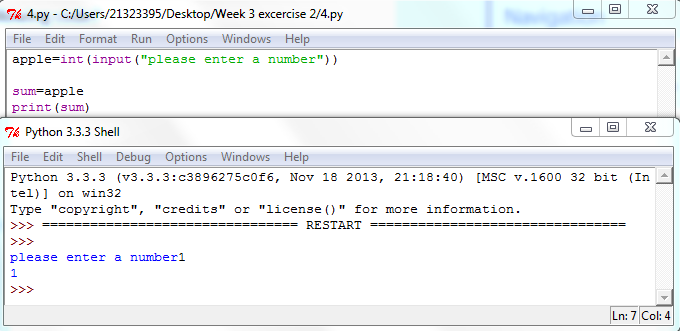
**All of the variable names below can be used. But which of these is the better variable name to use?**

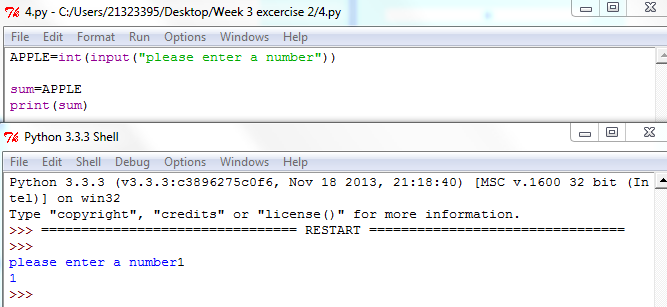
The better variable name to use from the choices given would in my opinion be the “areaOfRectangle” for the reason that the variable is now easily understood which is the point of naming the object we are representing.

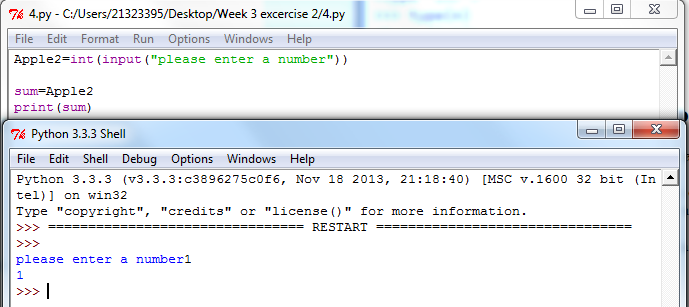
## Exercise 4

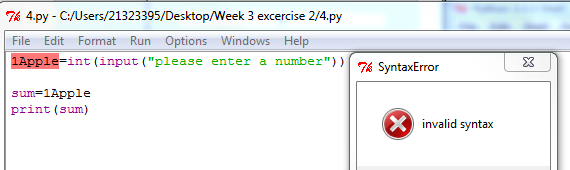
**Which of these variables names are not allowed in Python? (More than one might be wrong.)**

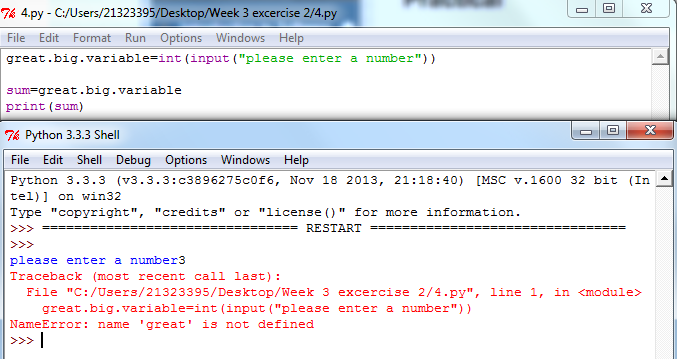
After testing all the variable names that were given to us I have come to the conclusion that the variables which are not allowed in Python are the ones starting with a number, for example: “1Apple”, “5Return”, the ones that have space in between the words, for example: “account number”, and lastly the ones with “.” In between the letters, for example: “great.big.variable”. Some of the variables I tested are shown below as snapshots. The rest of the variable names seem to be working and allowed in python or at least Python 3.5.

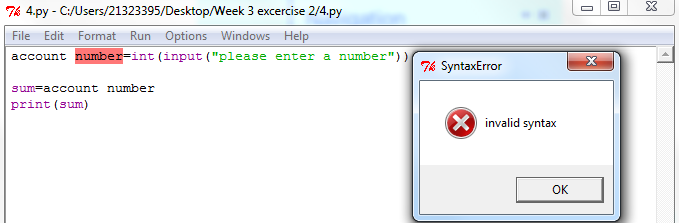


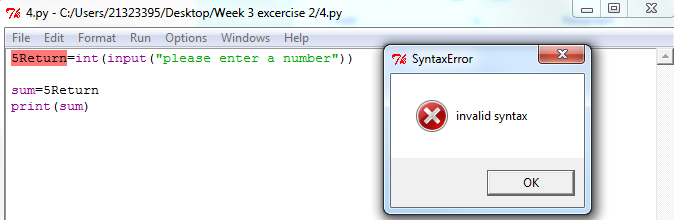












# Session 4

In this week’s lecture we discussed and talked about Python Data types, converting data types, Python operators along with some examples of the operators which made the subject much more understandable. Moreover we learned the order of operations that take place, some escape sequences with strings how to import modules and how to use them and also we talked about the programming errors that might come up when we are coding.

## Exercises 1

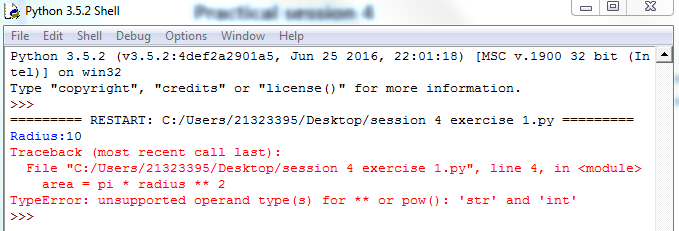
**Explain the mistake in the following code:**

**radius = input("Radius:")**

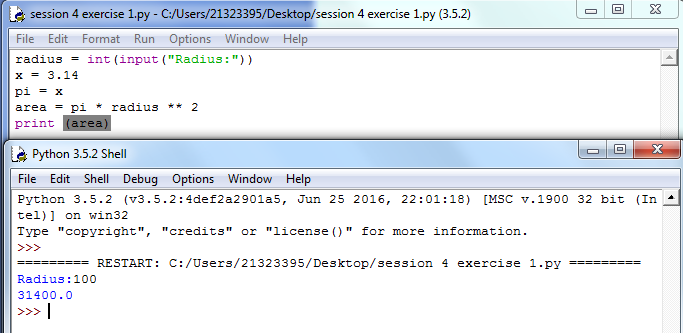
**x = 3.14**

**pi = x**

**area = pi \* radius \*\* 2**



When I run the code given it gives me an error. I believe it Is because I have to convert the radius into and integer.



I converted the radius to an integer and by doing that the compiler can now run the code and by using the “print” command thus the code is complete.

## Exercise 2

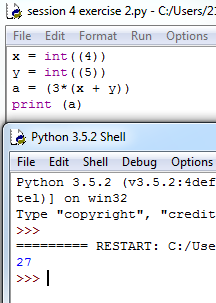
**Explain the mistake in the following code:**

**x = 4**

**y = 5**

**a = 3(x + y)**

The mistakes in the code was that the variables x and y were not converted to an integer and “3(x=y)” should be “(3\*(x+y))” for it to solve the equation. Moreover just as before I have to use the “print” command for the solution of the equation to be printed.

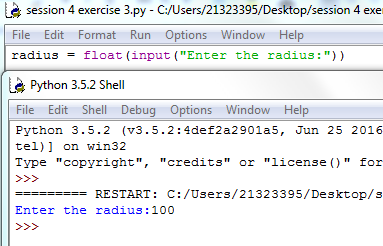


## Exercises 3

**Explain the mistake in the following code:**

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

The mistake of the above code is that instead of the code being “ float(input(“Enter the radius:”)) “, it is radius = float(input("Enter the radius:")).



## Exercises 4

**Why does this code not calculate the average?**

**print(3 + 4 + 5 / 3)**

The code is wrong because it is all in the same equation and according to the hierarchy of mathematics the number 5 is divided by 3 before the rest of the numbers are added up so the correct code would be:

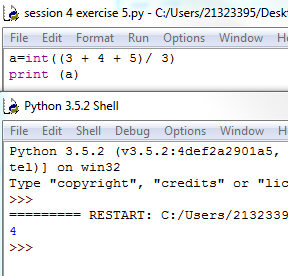
a=int((3+4+5)/3)

print=(a)

By doing this the parenthesis is calculated first and then divided by the amount of numbers which were added together which in this case is 3 so the process is as follows: 1)Add numbers (3,4,5)

2)Divide added amount by the amount of the numbers that were added together (3).

3)Print the solution to the equation (4).



## Exercise 5

**Consider the following code:**

**x = 19.93**

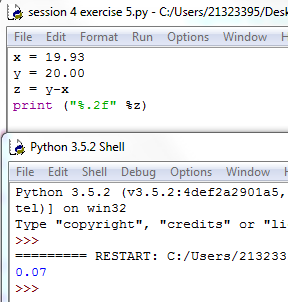
**y = 20.00**

**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.**



The command “%.2f” is a format string that basically tells the compiler to “print” the output to two decimal places so instead of the output being “0.0700000000028”, by using the command line “ print(%2.f” %z) we get “0.07” as the output.

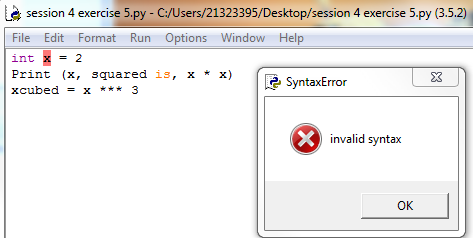
## Exercise 6

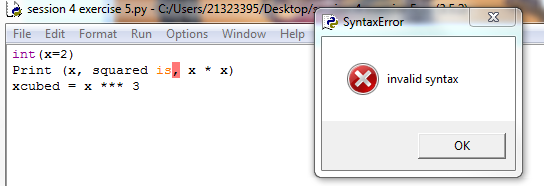
**Find at least three compile-time errors:**

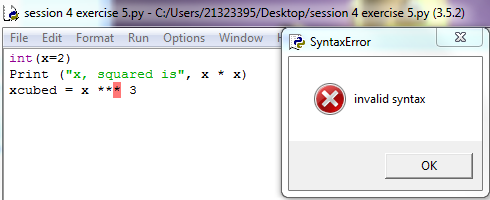
**int x = 2**

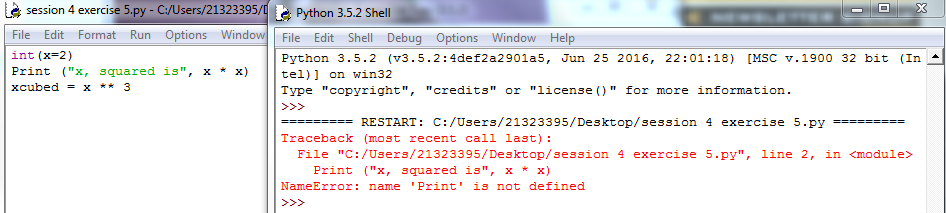
**Print (x, squared is, x \* x)**

**xcubed = x \*\*\* 3**









## Exercise 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))**

1. The code does not have the command “import math” for us to get something from that module.

2)The variables X,Y are in upper case letters but in the code lines print(“The product of “, x, “and”, y, “is”, x + y) and

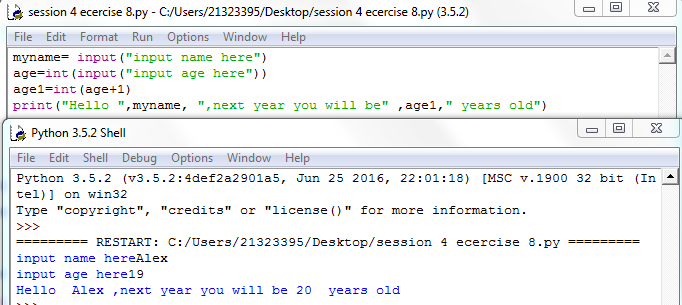
print(“The root of their difference is “, sqrt(x – y)) they are in lower case letters and as we’ve discussed in today’s lecture python code is case sensitive.

## Exercise 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!**

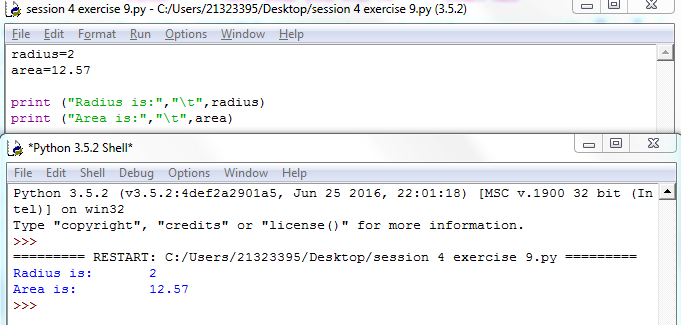


## Exercise 9

**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**

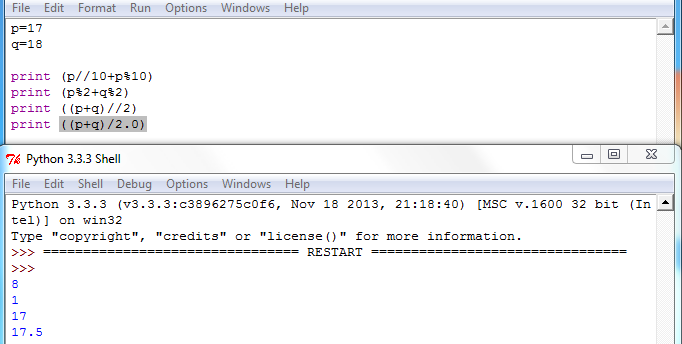
**Area is:**  **12.57**



## Exercise 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**



# Session 5

In this week’s session we talked about Flow Control Structures, Logical operators, Boolean Truth Tables and also discussed a few things about Hand Tracing.

## Exercise 1

**What is the error in this statement?**

**if scoreA = scoreB :**

**print("Tie")**

The error in this statement is the “=” which means assignment. The correct form of this statement should be “==” which means “equals”.

If scoreA == scoreB

Print(“Tie”)

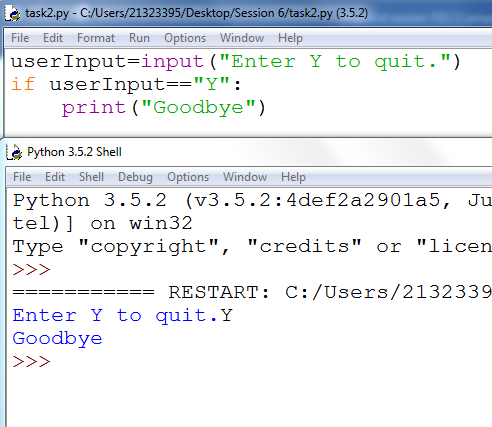
## Exercise 2

**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”**

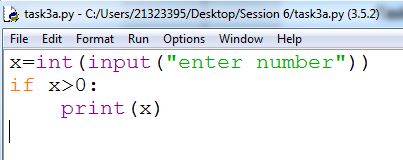


## Exercise 3

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

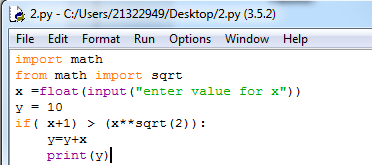
**a) if x > 0 then :**

**print(x)**



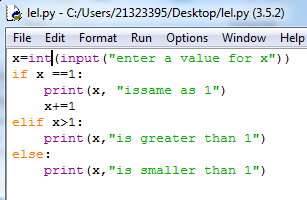
**b) if 1 + x > x \*\* sqrt(2) :**

**y = y + x**



**c) if x = 1 :**

**y += 1**



**d)**

**letterGrade = "F"**

**if grade >= 90 :**

**letterGrade = "A"**

**if grade >= 80 :**

**letterGrade = "B"**

**if grade >= 70 :**

**letterGrade = "C"**

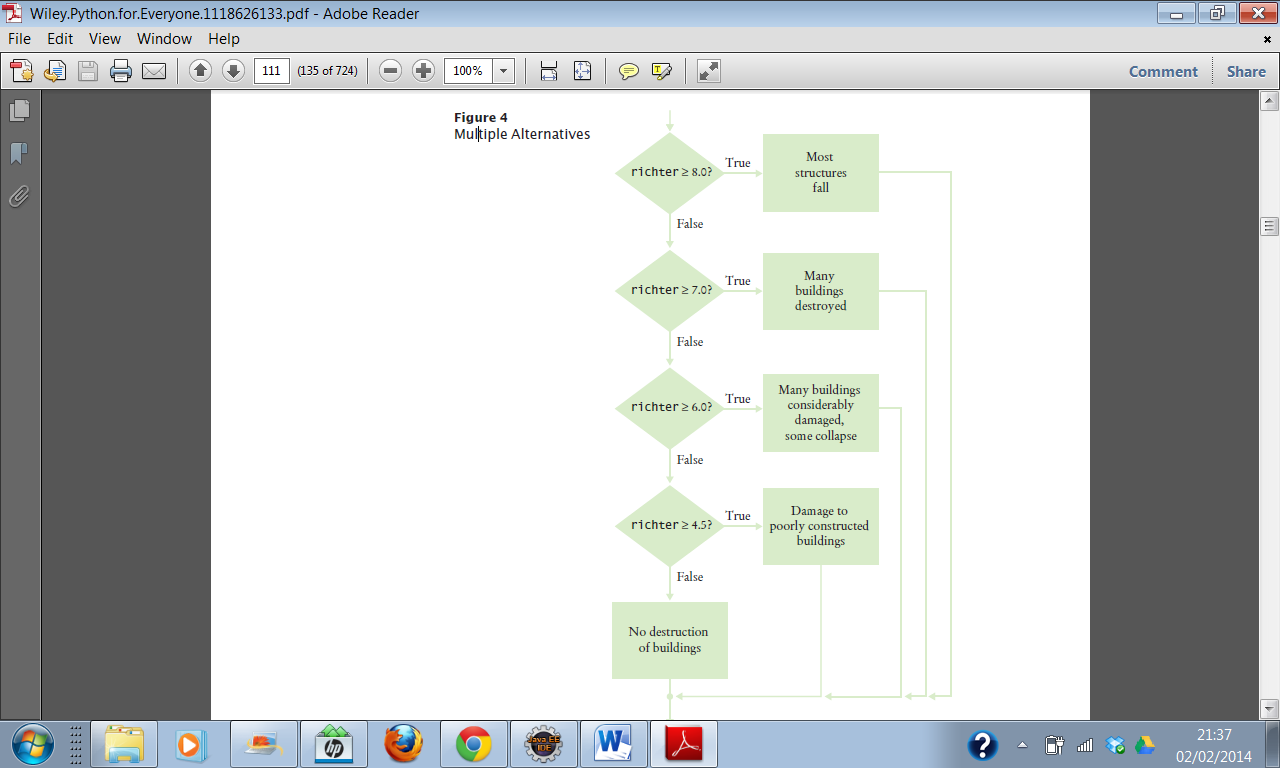
**if grade >= 60 :**

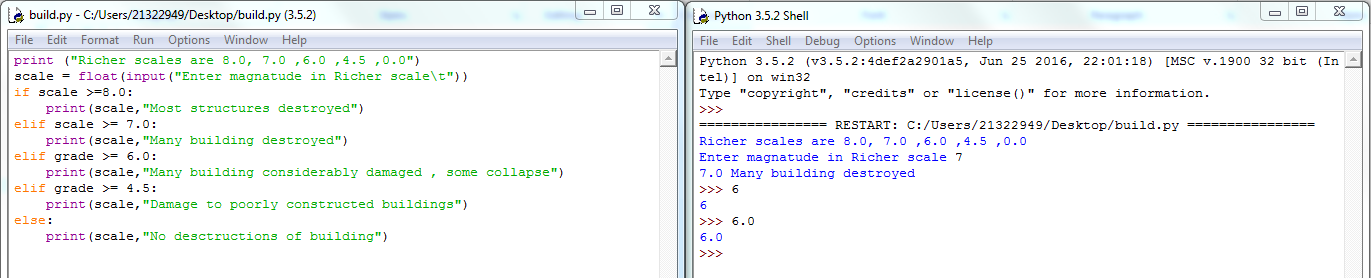
**letterGrade = "D"**

## Exercise 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”).**





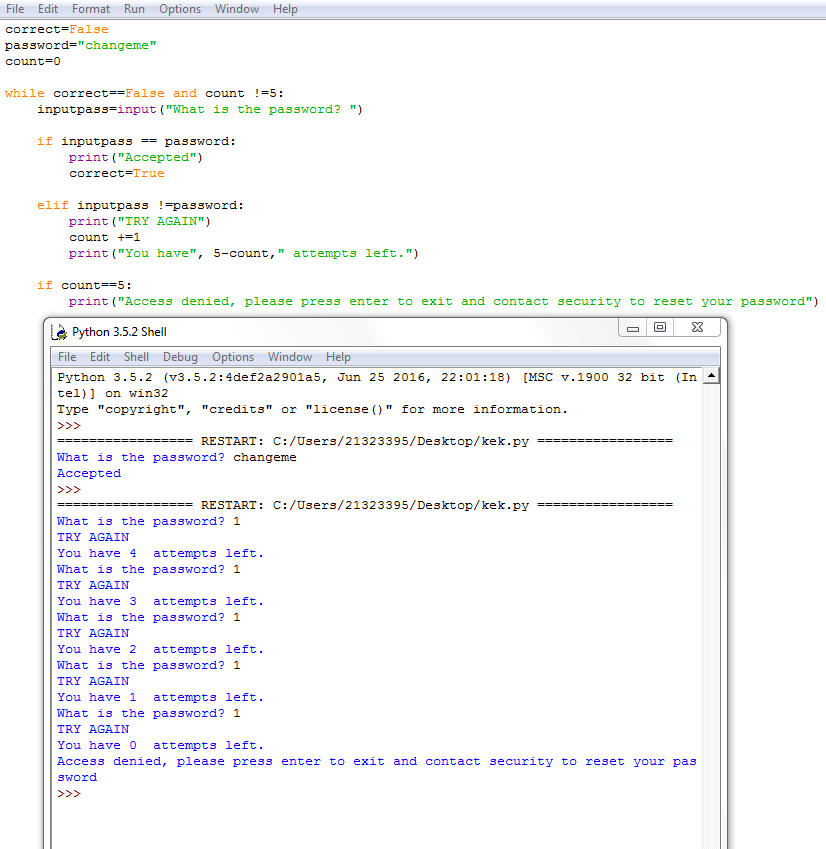
## Exercise 5

## **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”**



## Exercise 7

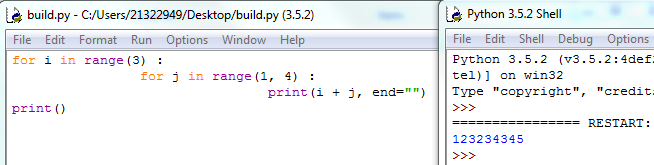
## **What do the following nested loops display? Hand trace.**

## **for i in range(3) :**

## **for j in range(1, 4) :**

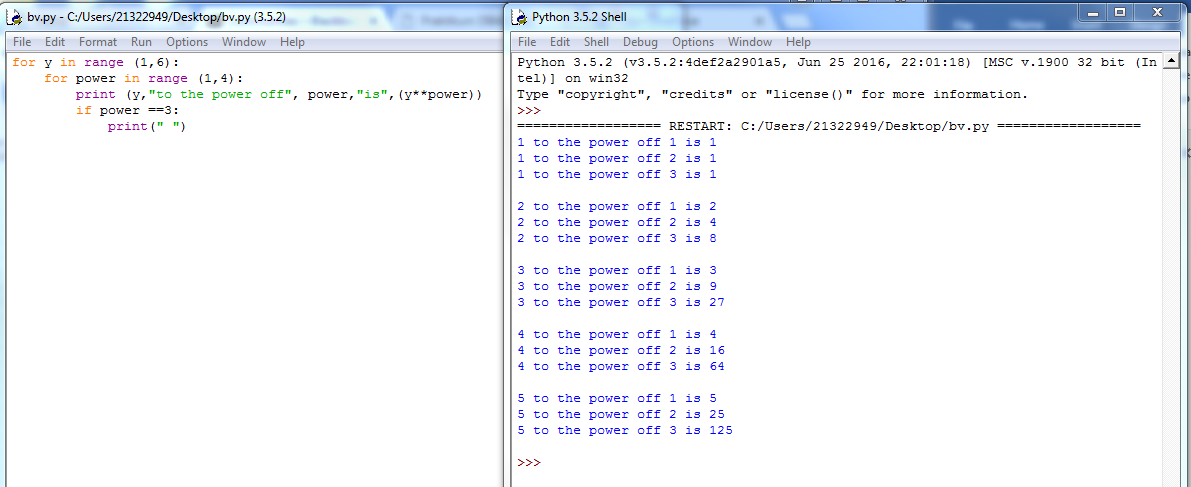
## **print(i + j, end="")**

**print()**



## Exercise 8

**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.**



# Session 6

In today’s session we made a brief revision of the Flow Control Structures and then started talking and learning about Data structures such as the main differences between each Data Structure and the common operations of these Data Structures.

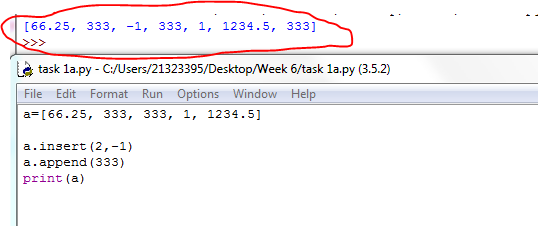
## Exercise 1

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

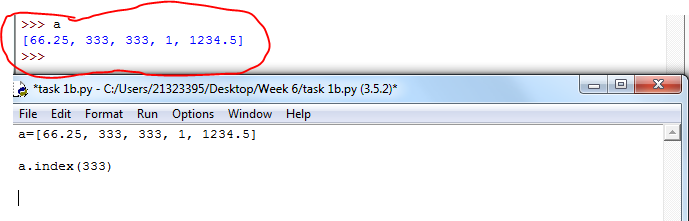
**If you perform the following operations on the list:**

1. a.insert(2, -1)

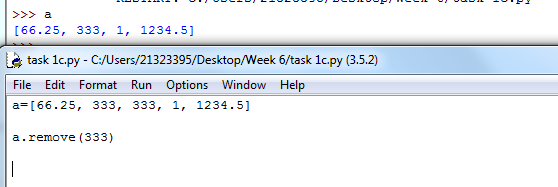
a.append(333)

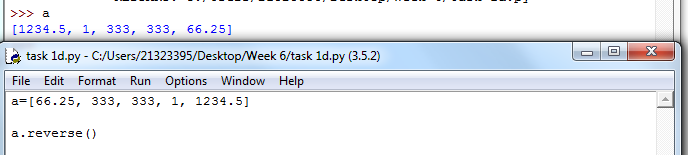


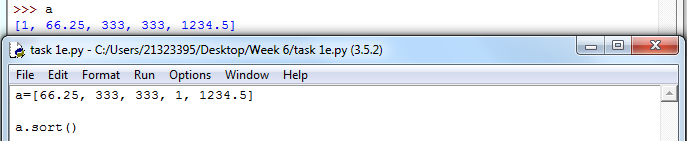
1. a.index(333)



1. a.remove(333)

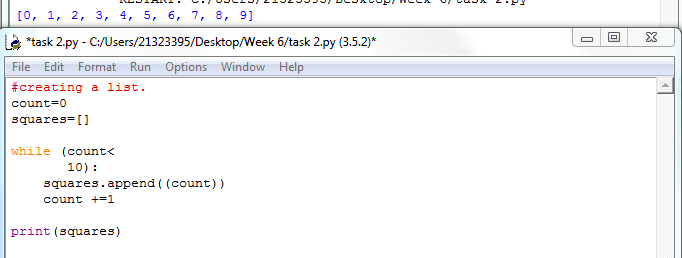


1. a.reverse()
2. a.sort()



## Exercise 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.**



## Exercise 3

**nums = []**

**for x in [1,2,3]:**

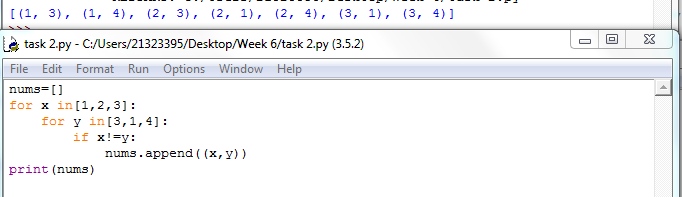
**for y in [3,1,4]:**

**if x != y:**

**nums.append((x, y))**

**print(nums)**

**What is the output from above?**

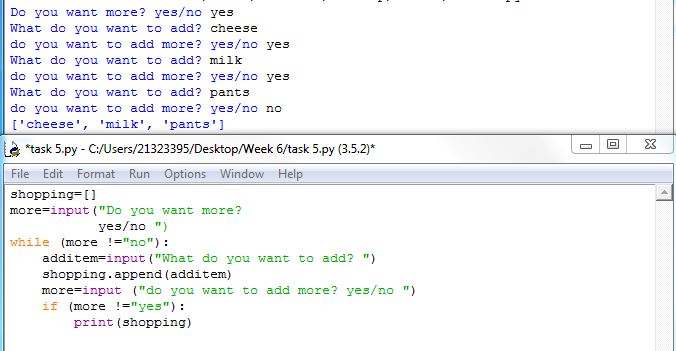


## Exercise 4

**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.**



## Exercise 5

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

**-** **Concatenate the two sentences into one long sentence**

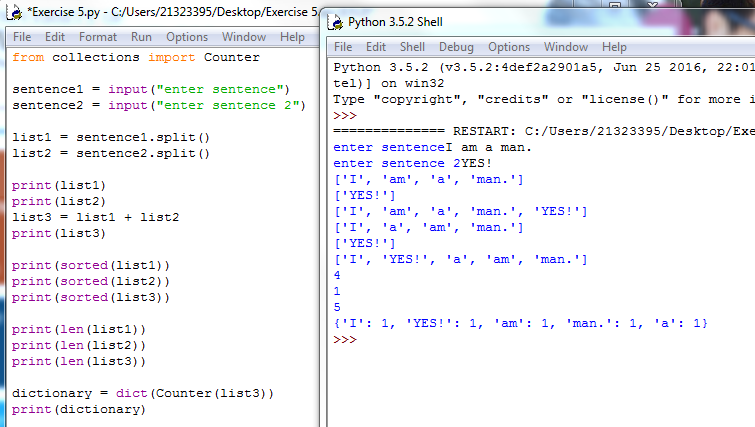
**-** **Split the sentence into a list of words**

**-** **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**



# Session 7 and 8

This week we discussed Functions and Methods and talked about how to implement the functions and were also shown some dictionary examples, how to call a function and what Return values are.

## Exercise 1

1. **Consider this function:**

**def mystery(x, y) :**

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

**return result**

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

The result of the function will be:

x=2

y=3

result = (2+3)/(3-2)

result = 5/1

result=5

## Exercise 2

**What does this program print?**

**def main() :**

**a = 5**

**b = 7**

**print(mystery(a, b))**

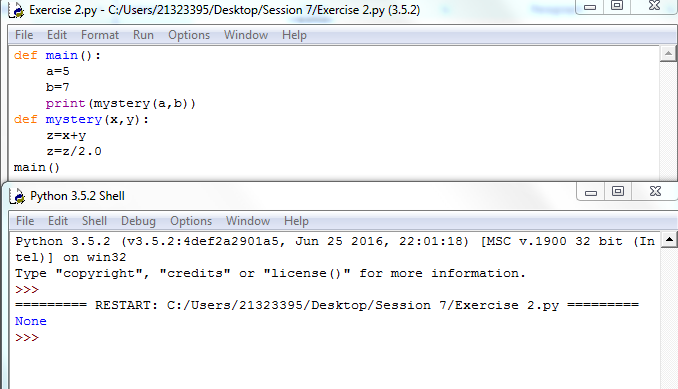
**def mystery(x, y) :**

**z = x + y**

**z = z / 2.0**

**return z**

**main()**



## Exercise 3

**What does this program print?**

def main() :

a = 4

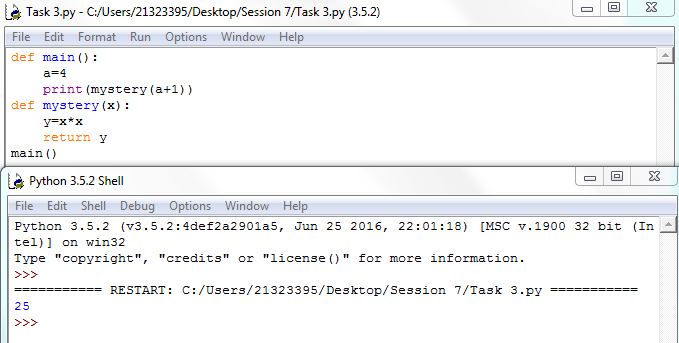
print(mystery(a + 1))

def mystery(x) :

y = x \* x

return y

main()



## Exercise 4

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

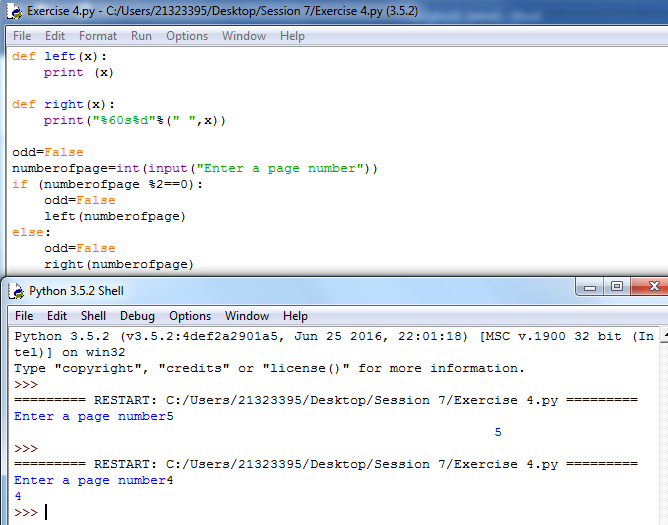
**if page % 2 == 0:**

**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.**



## Exercise 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.**

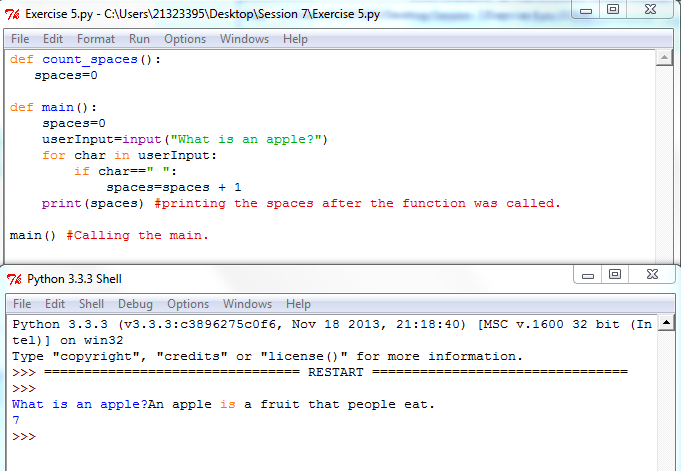
**# Counts the number of spaces**

**spaces = 0**

**for char in userInput :**

**if char == " " :**

**spaces = spaces + 1**



## Exercise 6

**Consider this recursive function:**

**def mystery(n) :**

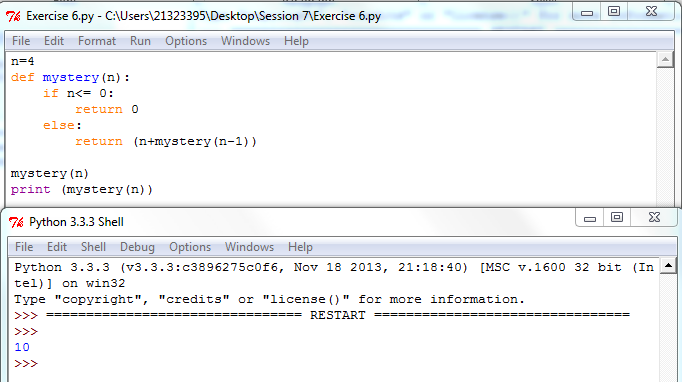
**if n <= 0 :**

**return 0**

**else:**

**return n + mystery(n - 1)**

**What is mystery(4)?**



## Exercise 7

**def mystery(n) :**

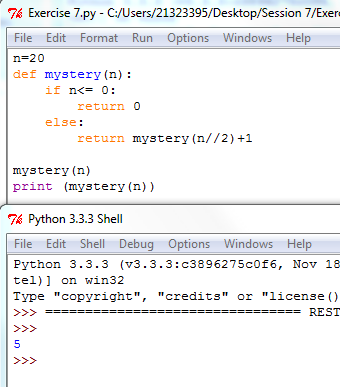
**if n <= 0 :**

**return 0**

**else:**

**return mystery(n // 2) + 1**

**What is mystery(20)?**



## Exercise 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)**

**def k(2) = 2\*(2+1) = 6**

**def h(2) = 2\*2+6-1= 9**

**def g(2) = 4 \* 9 = 36**

**def f(2) = 36 + √(9) = 39**

1. **x1= 39**
2. **x2 = 36**
3. **x3 = 92**
4. **k(0) = 2\*(0+1) =2**

**h(0) = (0\*0)+2-1= 1**

**g(0) = 4 \* 1 = 4**

**f(0) = 4+ √1 = 4+1 = 5**

**k(1) = 2 \* (1+1) = 4**

**h(1) = (1\*1) + 4 -1 =1 + 4 – 1 = 4**

**g(1) = 4 \* 4 = 16**

**f(1) = 16 + √4 = 16 + 2 = 18**

**k(2) = 2 \* (2 + 1) = 6**

**h(2) = (2 \* 2) + 6 – 1 = 9**

**g(2) = 4 \* 9 = 36**

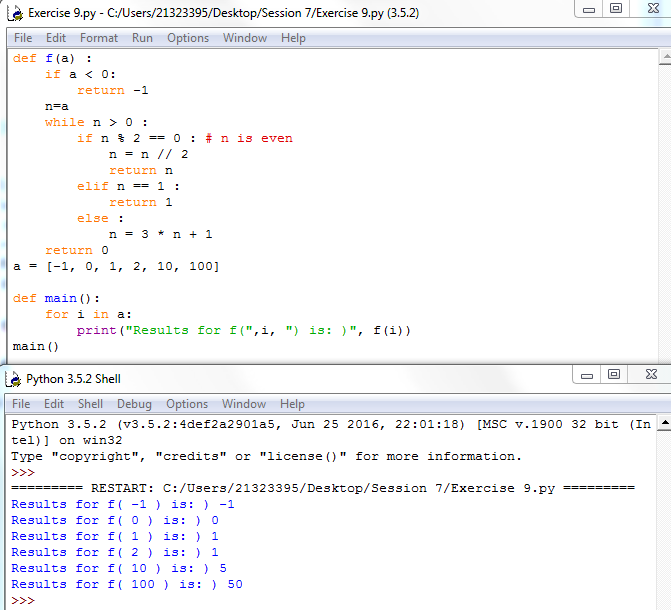
**f(2) = 36 + √9 = 36 + 3 = 39**

**x4 = 5 + 18 + 39 = 62**

1. **x5 = 0 + 0 + 0 +0 = 0**

## Exercise 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). |



# Session 9

In session 9 we had discussed Object Oriented Programming(OOP) and discussed what a “class” and “instance” is and what is the difference between them. Moreover, we were taught how to construct objects and classes.

## Exercise 1

Consider the class:

**class Person():**

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

**self.\_name = firstName**

If an object is constructed as

**harry = Person("Harry")** what is the value of its instance variable **\_name**?

harry.\_name = “Harry”

The value of the instance harry.name=”Harry” is a string.

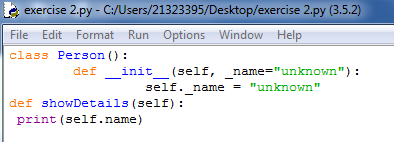
## Exercise 2

Provide an implementation for a Person constructor so that after the call p = Person() the **\_name** instance variable of p is "**unknown**"

**class Person():**

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

**self.\_name = “unknown”**



## Exercise 3

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

class **CashRegister** :

**# Comment 1**

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

*self*.\_itemCount = 0

*self*.\_totalPrice = 0.0

**# Comment 2**

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

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

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

**# Comment 3**

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

return *self*.\_totalPrice

**# Comment 4**

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

return *self*.\_itemCount

**# Comment 5**

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

*self*.\_itemCount = 0

*self*.\_totalPrice = 0.0

3a)

Comment 1: Keeps count of items and the total price of the items by using a constructor to initialize data fields.

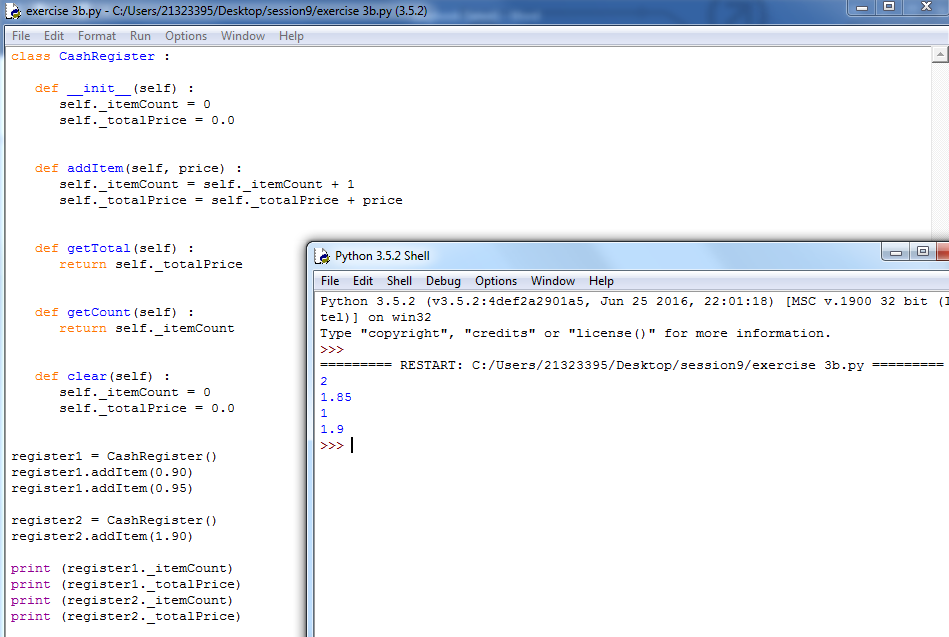
Comment 2: For each item added the program adds 1 to the item count and adds the price of the item to the total price.

Comment 3: The code here returns the sum of the total price to the user.

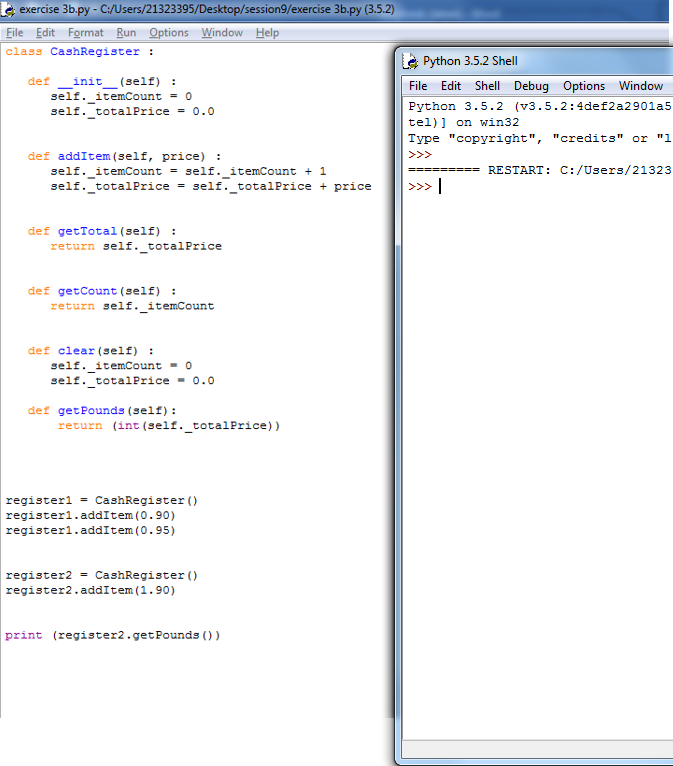
Comment 4: The code here returns the sum of the item count to the user.

Comment 5: The code here clears all of the previous data it had and makes everything 0.

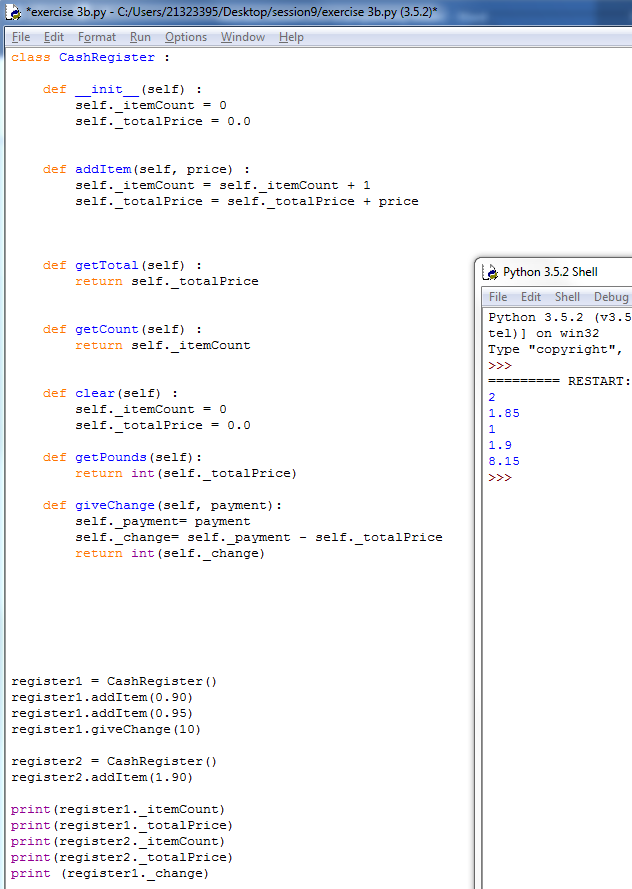
3b)



3c)



3d)



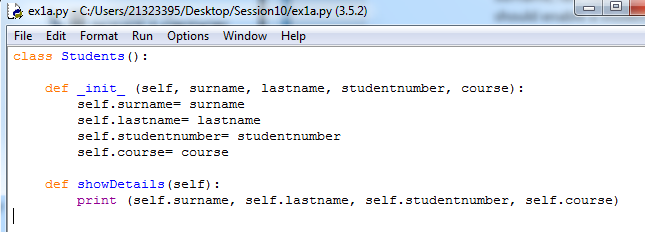
# Session 10

In session 10 our lecturer informed us about how to test a class and the difference between a “Method” and a “Function” and also what “Inheritance” is.

## Exercise 1

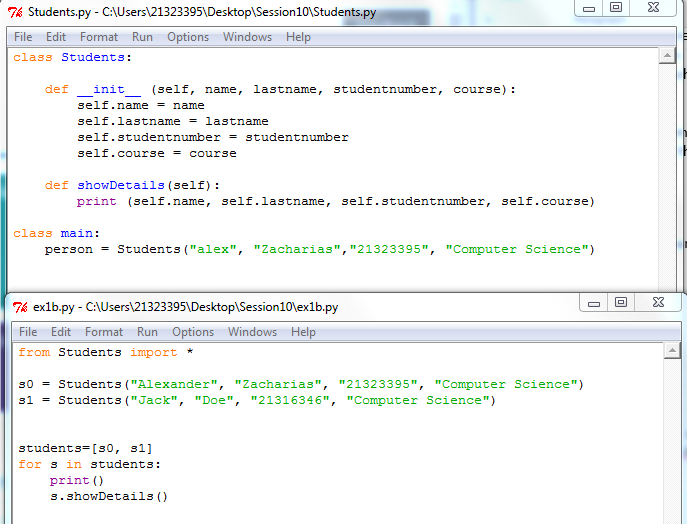
1a)

Write a class “Student” with a state that stores information about the students 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. #



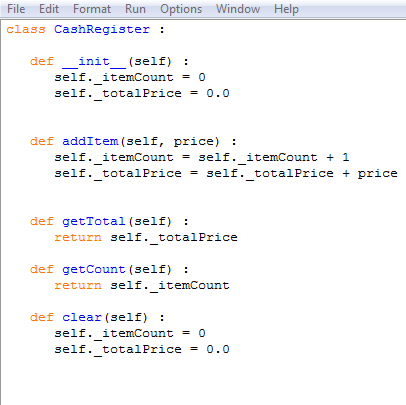
1b)

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



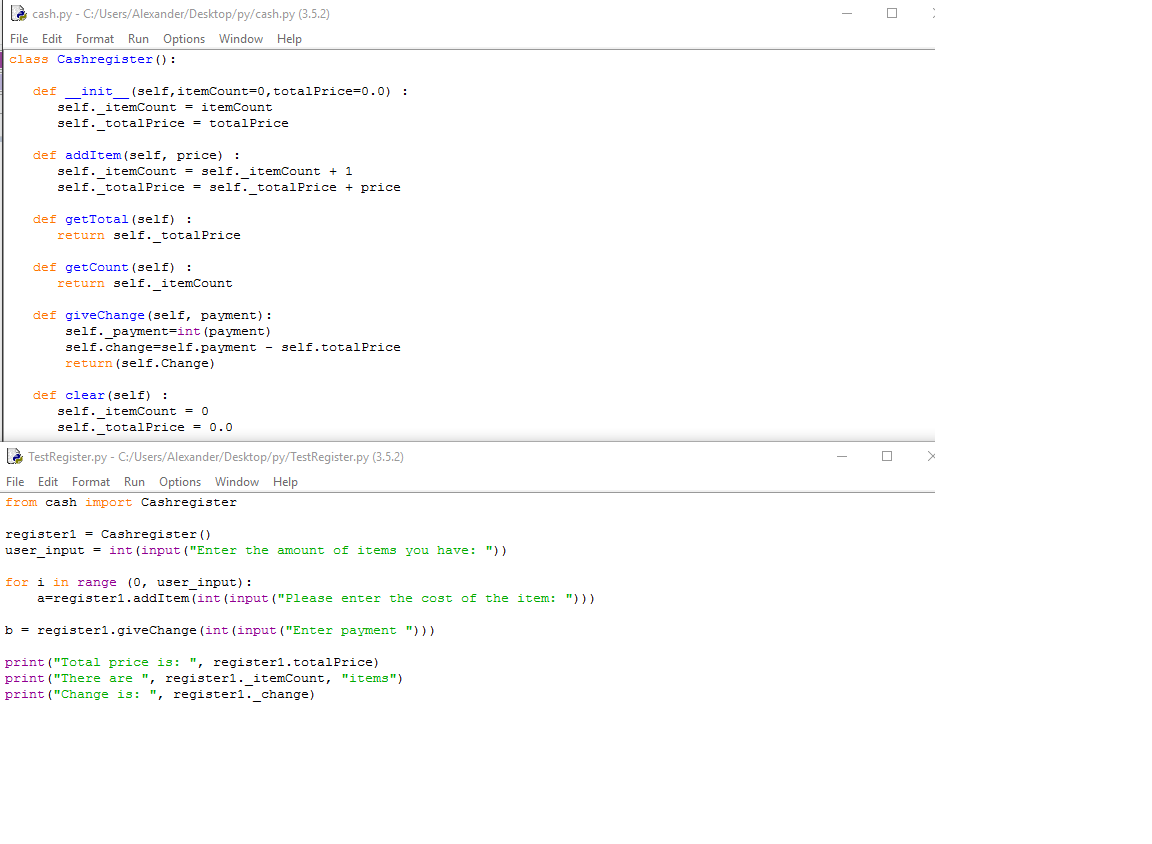
## Exercise 2

2a) A simulated cash register that tracks the item count and the total amount looks like this:



2b)

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



**Github:** <https://github.com/alexandernzach/Student-UWL>