# Department of Computing

# 

**CS114: Fundamentals of Programming**

**Class: BESE 9AB**

**Lab 02: Introduction of Python and IDLE (Integrated Development Environment)**

**CLO1: Understand the syntax and semantics of different programming constructs**

**Date: 14th September 2018**

**Time: 9:00am -12:00pm /02:00pm-05:00pm**

**Instructor: Ms. Hania Aslam**

## Lab 1: Introduction of Python and IDLE

### Introduction

This lab is designed to develop basic understanding with python and its development environment IDLE.

### Objectives

The objective of this exercise is to become familiar with the Python IDE for version 3.x while introducing basic mathematical operations, variable types, and printing options.

### Tools/Software Requirement

Python IDLE

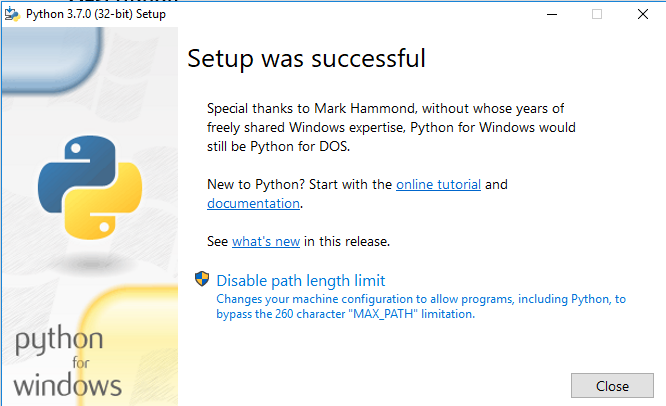
### Description

#### Installation:

Download the Python 3.x from this link: <https://www.python.org/downloads/>

[1 Mark]

|  |
| --- |
| Snap of Installation |
| Add snap here. |

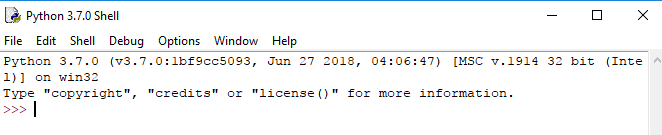


Once downloaded, double click on the .exe file downloaded and run the installation procedure.

You can select the IDLE (Python), by pressing windows key, and type IDLE, select IDLE (Python).

[1 Mark]

|  |
| --- |
| Snap of Python IDLE |
| Add snap here. |



#### Introduction to Python:

Python is a widely used [high-level programming language](https://en.wikipedia.org/wiki/High-level_programming_language) used for [general-purpose programming](https://en.wikipedia.org/wiki/General-purpose_programming_language), created by Dutch [programmer](https://en.wikipedia.org/wiki/Programmer) [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) and first released in 1991. Its syntax allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code) than possible in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B) or [Java](https://en.wikipedia.org/wiki/Java_%28programming_language%29).

Virtually all modern programming languages make use of an IDE, or Integrated Development Environment, which allows the creation, editing, testing, and saving of programs and modules. In Python, the IDE is called IDLE (like many items in the language, this is a reference to the British comedy group Monty Python, and in this case, one of its members, Eric Idle).

Before opening IDLE, it is worth recalling that there are three basic types of simple variables in Python: integers (whole numbers, commonly used as an index), floats (that is, numbers with a decimal point, AKA real numbers), and strings (collections of alphanumeric characters such as names, sentences, or numbers that are not manipulated mathematically such as a part number or zip code). A legal variable name must start with a letter. It is then optionally followed by some collection of letters, numerals and the underscore. It cannot contain any other characters or spaces, and cannot be a reserved word (i.e., a word with a special meaning in the language such as a command or operator). In Python, variables may be created by simply declaring them and assigning a value to them. Examples include:

a=2.3

name=”Joe”

It is best to think of the equal sign as “gets”. That is, think of the first example as “the variable a gets the floating point value 2.3” and the second as “the variable name gets the string Joe”. An assignment command such as these literally reserves space in the computer’s memory for the variable and tags it with the variable name. Then, it stores the appropriate value at that location for future use.

Procedure – Output Window Open IDLE by selecting Python from the Start menu and then choosing the option to open IDLE (Python GUI). Do NOT open the command line. Alternately, open the IDLE (Python GUI) icon on the desktop. A simple text window known as the Python shell will open. It should have a white background with a text message at the top and immediately below that, a cursor prompt >>> The shell serves two functions. First, it can serve as a sort of scratch pad to try snippets of code (shown in the steps below). Second, it can serve as the text output window for larger programs. Do not try to use this window for the creation of complete programs that you wish to save. We shall use this window to create a few variables and perform some basic manipulations on them. Type the following and then hit the Enter key: a=5 The >>> should reappear. This command defines a variable called a and copies the integer value 5 into it. In similar manner, type in the following commands: b=13 x=5.0 y=13.0 m=”Mary” n=”Nancy” It is very important that the “.0” portions be included. This is how integers and floats are distinguished: floats always have a decimal point, integers don’t. Also, it is possible to define the strings using the apostrophe‘ versus the quote ”. This can be handy if you need to have a string that includes a quote or apostrophe within it; merely define the string with the other character. In any case, the computer’s memory now looks something like this:

|  |  |
| --- | --- |
| Name | Value |
| a | 5 |
| b | 13 |
| x | 5.0 |
| y | 13.0 |
| m | Marry |
| n | Nany |

The trick now, of course, is to access these values, manipulate them, and see the results. An important command for this process is the print command. print will print what follows it, either variables or expressions, on to the output window. Note that like all built-in commands and functions in Python, this command is all lower case. Capitalizing it will generate an error. Also, note that commands will be color coded orange-red.

At the prompt, type the following:

print ("hello world")

The output should be hello world

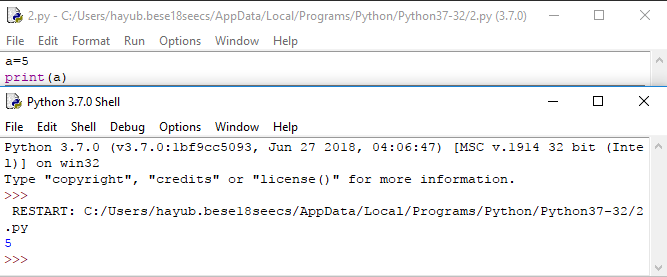
Now type:

**print( a )**

The output should be the integer 5

[1 Mark]

|  |
| --- |
| Output of print(a) |
| Add snap here. |



Now type:

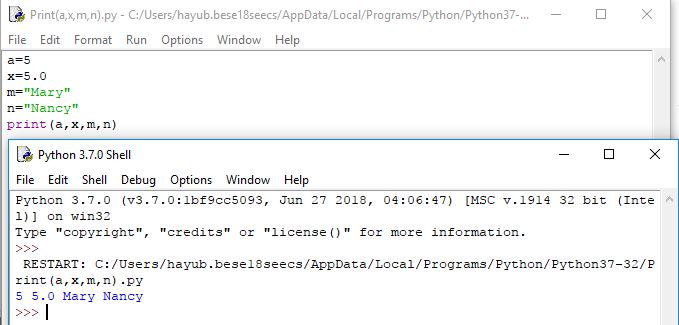
**print( a, x, m, n )**

In this case, the following sequence should result:

5 5.0 Mary Nancy

[1 Mark]

|  |
| --- |
| Output of print(a, x, m, n) |
| Add snap here. |

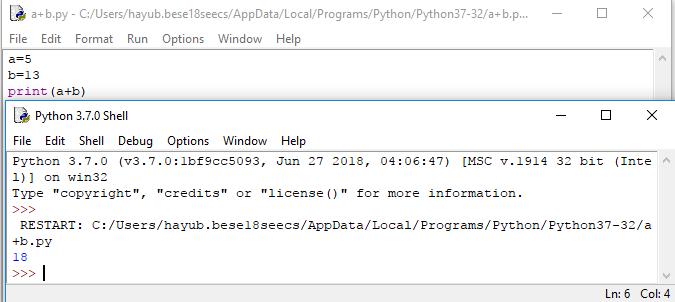


Continue with the following expression:

**print( a + b )**

[1 Mark]

|  |
| --- |
| Output of print(a+b) |
| Add snap here. |



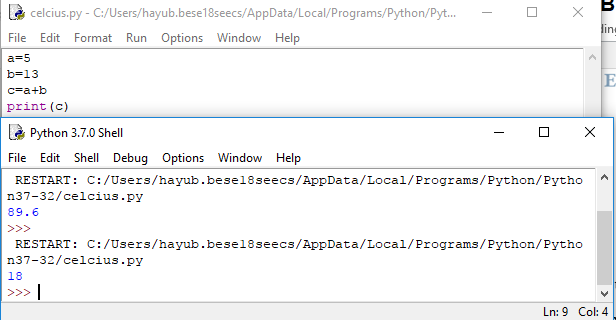
This results in the value 18. This line retrieves the values of a and b from memory, adds them together, and prints the result on the output window. Neither a nor b are altered in the process. Alternately, we could have created a brand new variable and printed it. The result will be the same. Enter the following two lines to verify this:

c = a + b

**print( c )**

[1 Mark]

|  |
| --- |
| Output of print(c) |
| Add snap here. |



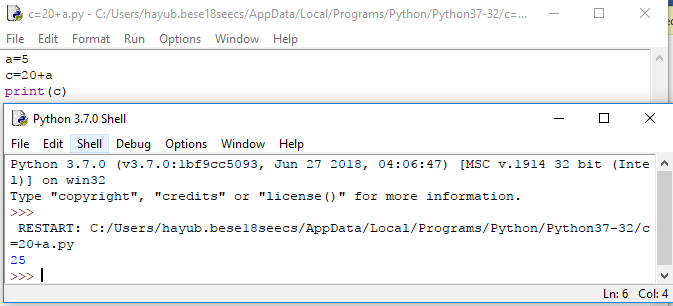
The only difference is that this version adds a new “slot” called c to the memory map above. It is worth noting that once a variable is created, its value may be recomputed over and over if desired. For example, type the following:

c = 20 + a

**print( c )**

[1 Mark]

|  |
| --- |
| Output of print(c) |
| Add snap here. |



The result should be 25. The first line computes a new value which then overwrites the prior value of 18.

Besides addition, the other main math operators are -, \* (multiplication), / (division), \*\* (exponents, which can also be performed using the function pow(x,y) for xy, % (modulo), and // (floor divide). Parentheses () may be used to force the execution of some operations before others. Parentheses have the highest precedence and are followed by multiplication, division, addition and subtraction. That is, the expression a=b+c\*d will multiply c by d before b is added. To force the addition first, use parentheses: a=(b+c)\*d Remember, think of the equal sign as “gets” as in “a gets the value computed by…”. It is an assignment, not a true mathematical relation. That is, if at some point in the future the value of b was to change, a will not automatically be altered to reflect that change. This allows you to do the following:

c = c + 1

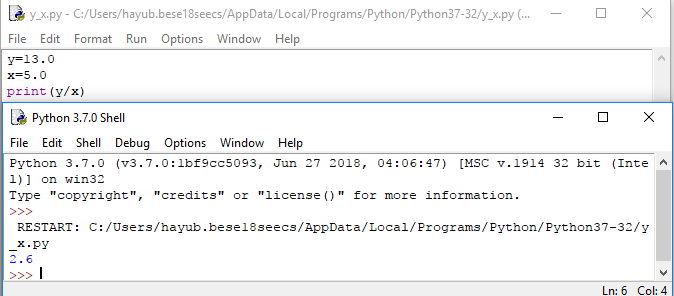
Type this in. What do you think the result will be?

The line above may appear a little odd. After all, how can something equal itself plus one? Remember, this is an assignment, not a mathematical relation. What it says is, “Retrieve the current value of c, add one to it, and store the result back in c (overwriting the original value). Print out the value of c. You should get 26 (the prior value of 25 plus one). Continuing with the other math operators, type:

**print( y/x )**

[1 Mark]

|  |
| --- |
| Output of print(y/x) |
| Add snap here. |

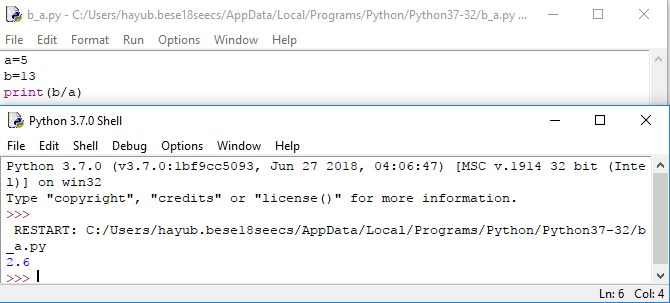


The result should be 2.6. Now try the following:

**print( b/a )**

[1 Mark]

|  |
| --- |
| Output of print(b/a) |
| Add snap here. |

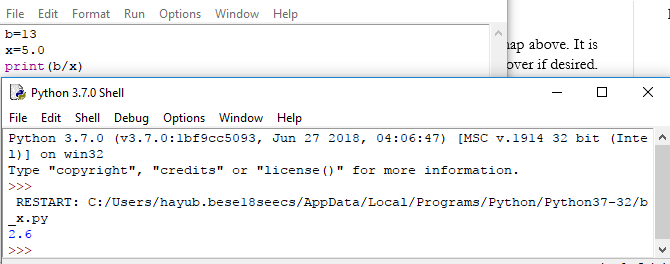


The result is also 2.6 even though both variables are integers (an integer, of course, can’t contain a fractional portion). In essence, Python promotes the variables to floats in order to maintain precision, producing a floating point answer. Now try:

**print( b/x )**

[1 Mark]

|  |
| --- |
| Output of print(b/x) |
| Add snap here. |



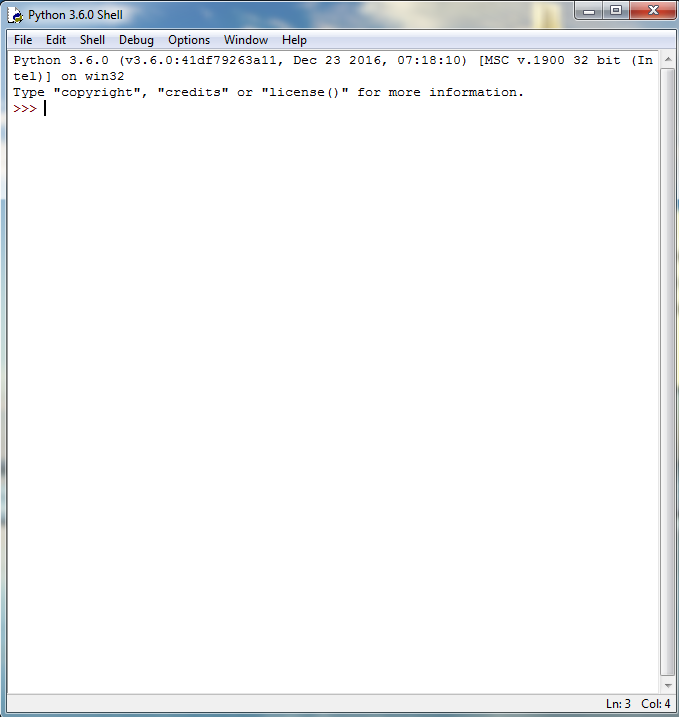
In this case the answer is again 2.6. This is because in a mixed calculation between a float and an integer, the integer is again promoted to a float in the calculation in order to maintain the precision of the floating point variable. You can force a variable to be promoted (or demoted) by using the float() and int() functions.

**Interactive Mode vs. Scripted Mode**

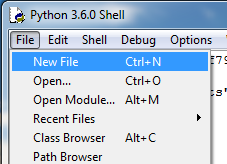
Python can be run in one of two modes. It can either be used interactively as explained above, via an interpreter, or it can be called from the command line/Scripted Mode to execute a script.

### To Open Python in Scripted Mode

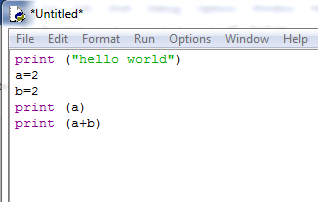
1. Open IDLE



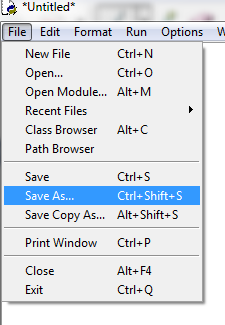
1. Go to *File* >> *New File*



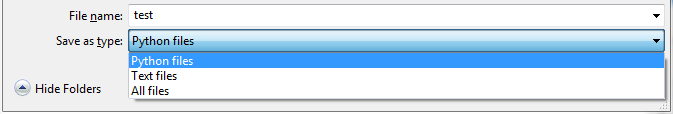
1. Enter your code



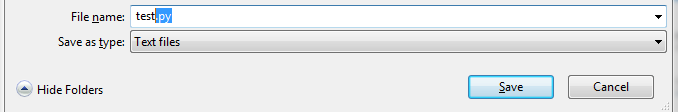
1. Save the file on your desired location



1. Either save the file by selecting *Python files* from *Save as type*



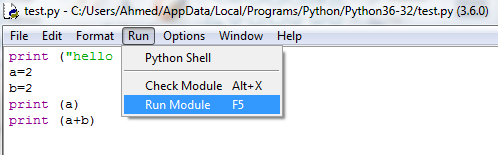
OR explicitly define the type of file by writing *.py* extension at the end of the name



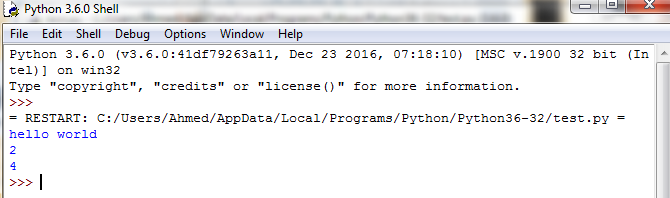
1. See the name of the file has been changed with the *.py* extension



1. Now Go to *Run* >> *Run Module* OR enter *F5* on your keyboard to run the code in Scripted Mode



1. See the output



**Final Lab Tasks:**

**[5 marks]**

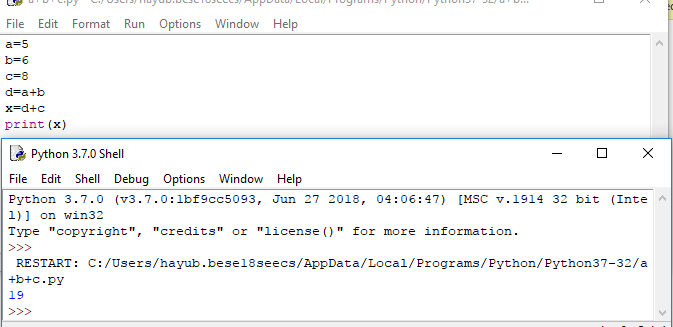
1. Write a program in the scripted mode that adds three numbers (of your choice) together. Use variables to store the values in your program and consider the following rules.

**Condition:** You cannot add more than two numbers in a single line i.e.

a+b+c is not allowed

a+b is fine.

|  |
| --- |
| Source Code+ Output Snap |
| Add your source code here. |

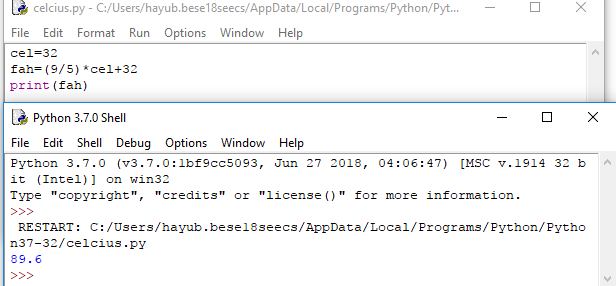


1. Write a program convert.py that converts the temperature from Celsius to Fahrenheit.

Conversion Formula: **(9/5) \* Celsius + 32**

**[5 marks]**

|  |
| --- |
| Source Code( convert.py)+ Output Snap |
| Add your source code here. |

****

**Deliverables**

Compile a single Word document by filling in the solution/answer part (as directed) along with the snapshots and source code. Name your submission file as given below and submit this Word file on LMS before the deadline.

**Name – Registration No. – Section**

**Name: Hamid Ayub**

**Registration no: 12933118**

**Section: SE\_9.B**

**Grade Criteria**

This lab is graded. Min marks: 0. Max marks: 20.

|  |  |  |
| --- | --- | --- |
| **Activity** | **Minimum** | **Maximum** |
| Documentation with clearly defined understanding of the lab task and approach | Fail | Pass |
| Installation | 0 | 01 |
| Python IDLE | 0 | 01 |
| print(a) | 0 | 01 |
| print( a, x, m, n ) | 0 | 01 |
| print( a + b ) | 0 | 01 |
| print( c ) | 0 | 01 |
| After adding 20, print( c ) | 0 | 01 |
| print( y/x ) | 0 | 01 |
| print( b/a ) | 0 | 01 |
| print( b/x ) | 0 | 01 |
| Lab Task 1 | 0 | 05 |
| Lab Task 2 | 0 | 05 |