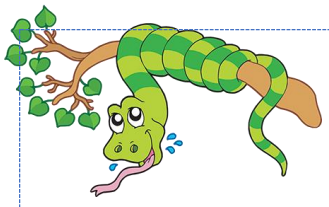


Python

Basics

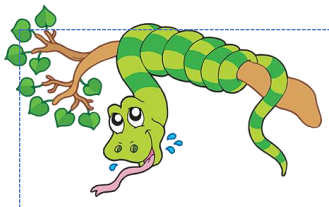
(Data Types, Variable)

Class viii

Lab 18

UNITED TRUST
ADD STRENGTHS TO THE WEAK

Powered By
bdeducation
Bangladesh Digital Education Research Limited
সময়ের পাঠশালা



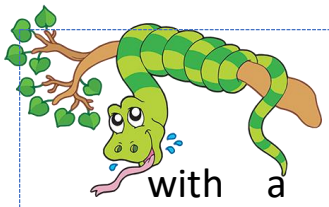
Lab Objectives:

- Integer, Floating-Point, String Data Types
- String Concatenation & Replication
- Storing Values in Variables

The Integer, Floating-Point, and String Data Types

Remember that expressions are just values combined with operators, and they always evaluate down to a single value. A data type is a category for values, and every value belongs to exactly one data type. The most common data types in Python are listed in "Table 2." The values -2 and 30, for example, are said to be integer values. The integer (or int) data type indicates values that are whole numbers. Numbers

Data type	Examples
Integers	`-2, -1, 0, 1, 2, 3, 4, 5`
Floating-point numbers	`-1.25, -1.0, --0.5, 0.0, 0.5, 1.0, 1.25`
Strings	`'a', 'aa', 'aaa', 'Hello!', '11 cats'`



with a decimal point, such as 3.14, are called floating-point numbers (or floats). Note that even though the value 42 is an integer, the value 42.0 would be a floating-point number.

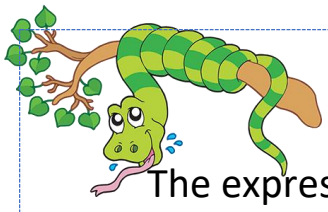
Python programs can also have text values called strings, or strs (pronounced “stirs”). Always surround your string in single quote (') characters (as in 'Hello' or 'Goodbye cruel world!') so Python knows where the string begins and ends. You can even have a string with no characters in it, '', called a blank string.

If you ever see the error message SyntaxError: EOL while scanning string literal, you probably forgot the final single quote character at the end of the string, such as in this example:

String Concatenation and Replication

The meaning of an operator may change based on the data types of the values next to it. For example, + is the addition operator when it operates on two integers or floating-point values. However, when + is used on two string values, it joins the strings as the string concatenation operator. Enter the following into the interactive shell:

```
>>> 'Alice' + 'Bob'
'AliceBob'
```



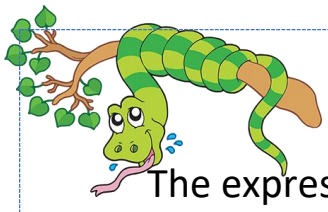
The expression evaluates down to a single, new string value that combines the text of the two strings. However, if you try to use the + operator on a string and an integer value, Python will not know how to handle this, and it will display an error message.

```
>>> 'Alice' + 42
Traceback (most recent call last):
  File "<pyshell#26>", line 1, in <module>
    'Alice' + 42
TypeError: Can't convert 'int' object to str implicitly
```

The error message Can't convert 'int' object to str implicitly means that Python thought you were trying to concatenate an integer to the string 'Alice'. Your code will have to explicitly convert the integer to a string, because Python cannot do this automatically. (Converting data types will be explained later when talking about the str(), int(), and float() functions.)

The * operator is used for multiplication when it operates on two integer or floating-point values. But when the * operator is used on one string value and one integer value, it becomes the string replication operator. Enter a string multiplied by a number into the interactive shell to see this in action.

```
>>> 'Alice' * 5
'AliceAliceAliceAliceAlice'
```



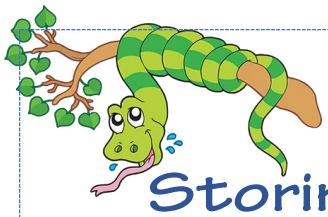
The expression evaluates down to a single string value that repeats the original a number of times equal to the integer value. String replication is a useful trick, but it's not used as often as string concatenation.

The * operator can be used with only two numeric values (for multiplication) or one string value and one integer value (for string replication). Otherwise, Python will just display an error message.

```
>>> 'Alice' * 'Bob'
Traceback (most recent call last):
  File "<pyshell#32>", line 1, in <module>
    'Alice' * 'Bob'
TypeError: can't multiply sequence by non-int of type 'str'

>>> 'Alice' * 5.0
Traceback (most recent call last):
  File "<pyshell#33>", line 1, in <module>
    'Alice' * 5.0
TypeError: can't multiply sequence by non-int of type 'float'
```

It makes sense that Python wouldn't understand these expressions: You can't multiply two words.



Storing Values in Variables

A variable is like a box in the computer's memory where you can store a single value. If you want to use the result of an evaluated expression later in your program, you can save it inside a variable.

Assignment Statements

You'll store values in variables with an assignment statement. An assignment statement consists of a variable name, an equal sign (called the assignment operator), and the value to be stored. If you enter the assignment statement spam = 42, then a variable named spam will have the integer value 42 stored in it.

Think of a variable as a labeled box that a value is placed in, as in Figure 1

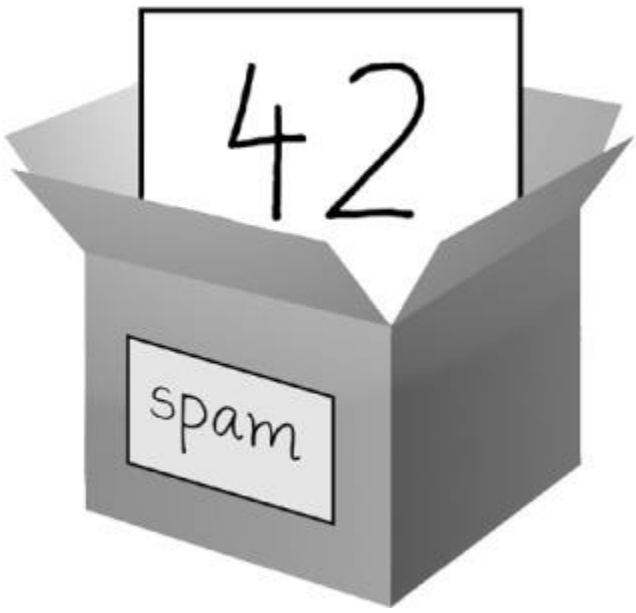
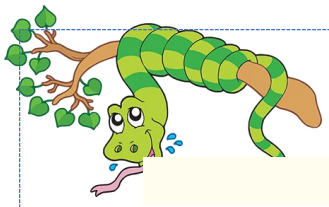


Figure 1

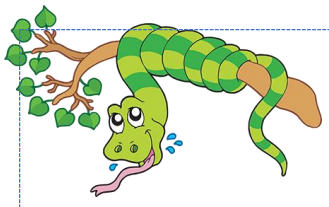


spam = 42 is like telling the program, “The variable spam now has the integer value 42 in it.”

For example, enter the following into the interactive shell:

```
❶ >>> spam = 40
  >>> spam
  40
  >>> eggs = 2
❷ >>> spam + eggs
  42
  >>> spam + eggs + spam
  82
③ >>> spam = spam + 2
  >>> spam
  42
```

A variable is initialized (or created) the first time a value is stored in it ❶. After that, you can use it in expressions with other variables and values ❷. When a variable is assigned a new value ③, the old value is forgotten, which is why spam evaluated to 42 instead of 40 at the end of the example. This is called overwriting the variable. Enter the following code into the interactive shell to try overwriting a string:



```
>>> spam = 'Hello'
>>> spam
'Hello'
>>> spam = 'Goodbye'
>>> spam
'Goodbye'
```

Just like the box in Figure 2, the spam variable in this example stores 'Hello' until you replace it with 'Goodbye'.

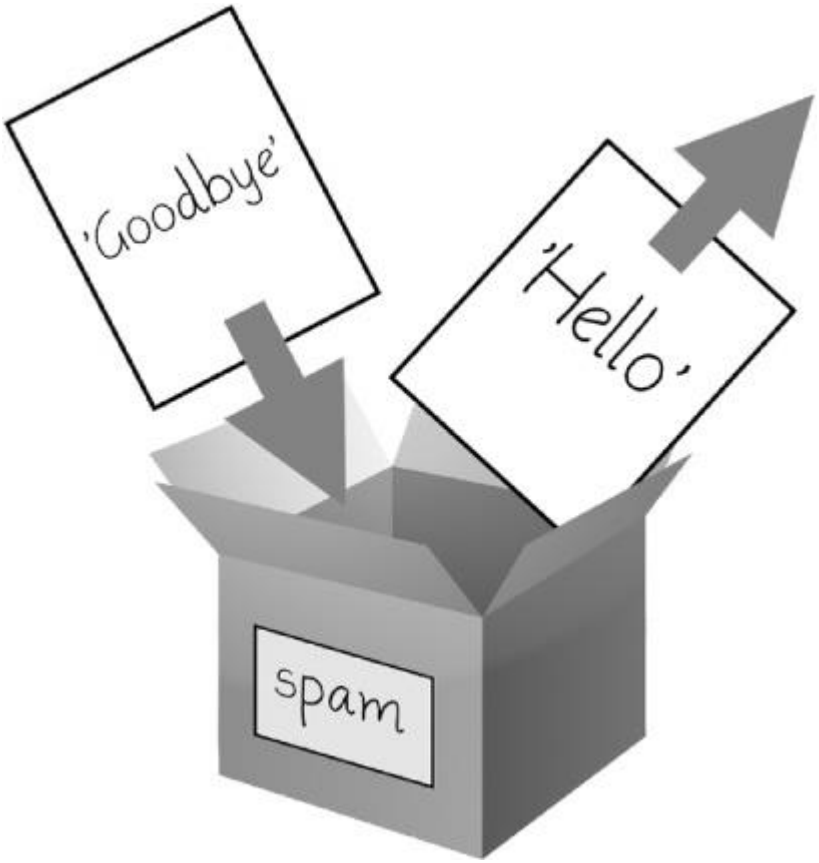
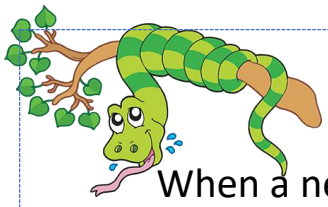


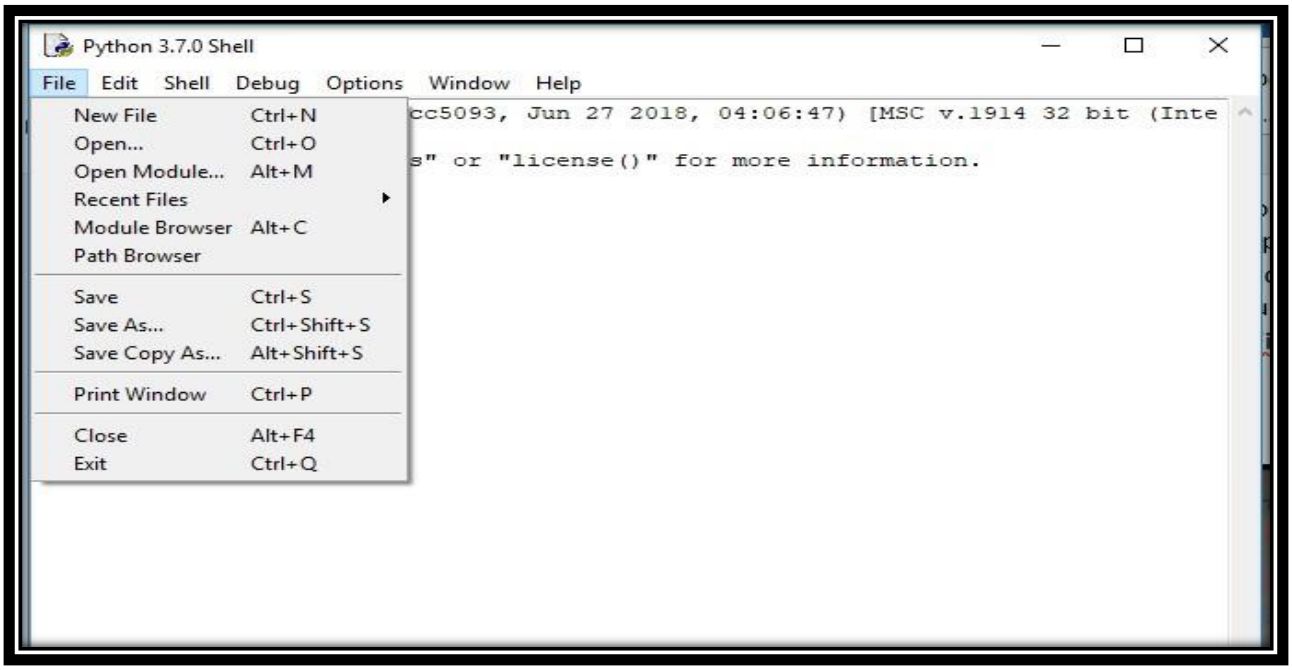
Figure 2



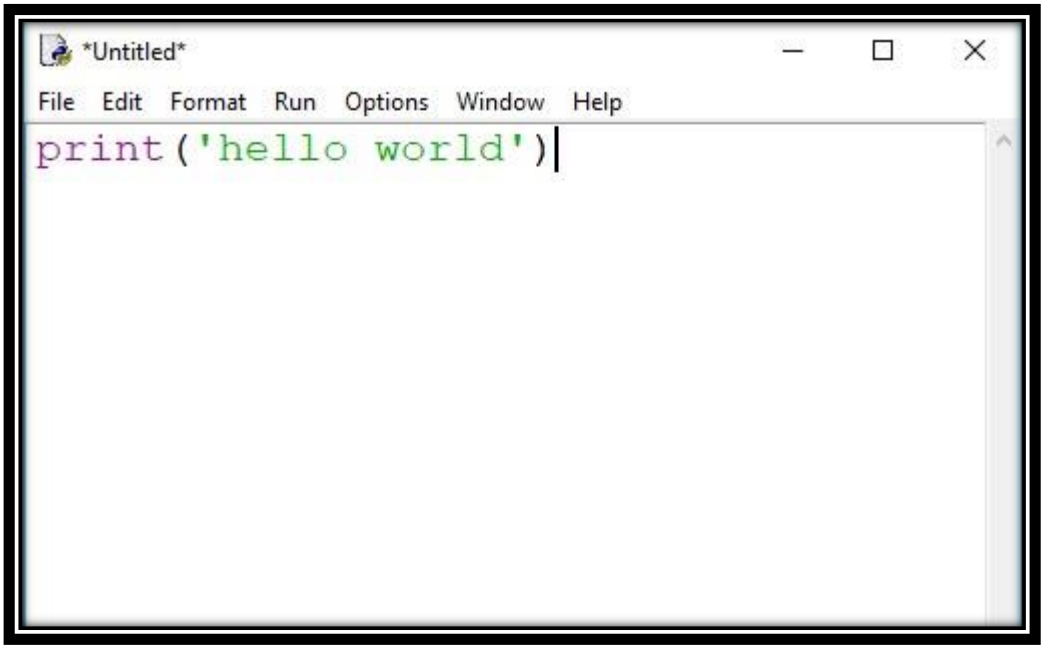
When a new value is assigned to a variable, the old one is forgotten.

Your First Program

While the interactive shell is good for running Python instructions one at a time, to write entire Python programs, you'll type the instructions into the file editor. The file editor is similar to text editors such as Notepad or TextMate, but it has some specific features for typing in source code. To open the file editor in IDLE, select File ▸ New File.

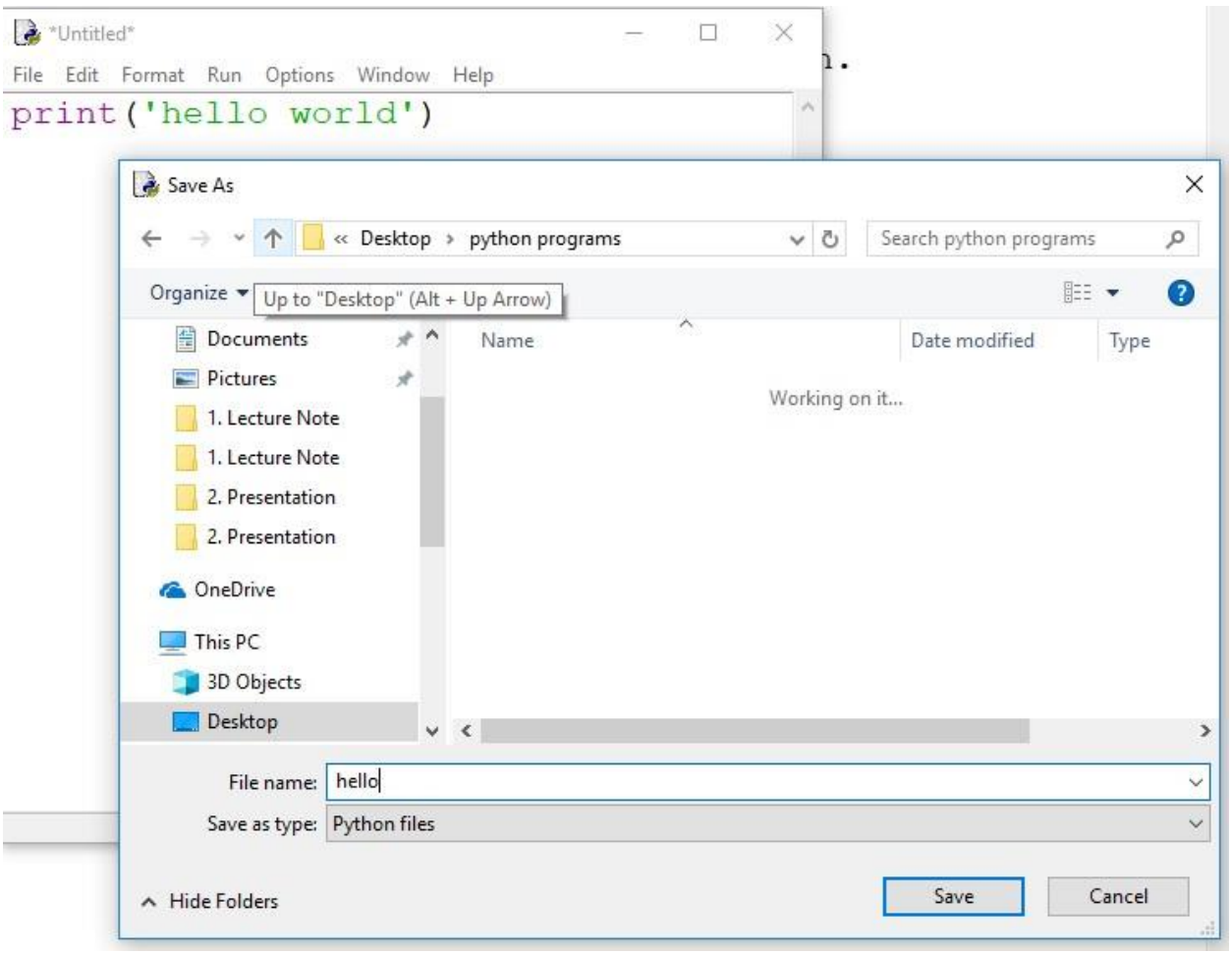


Python file editor will open in another tab. Now type *Print 'hello world'* in it.

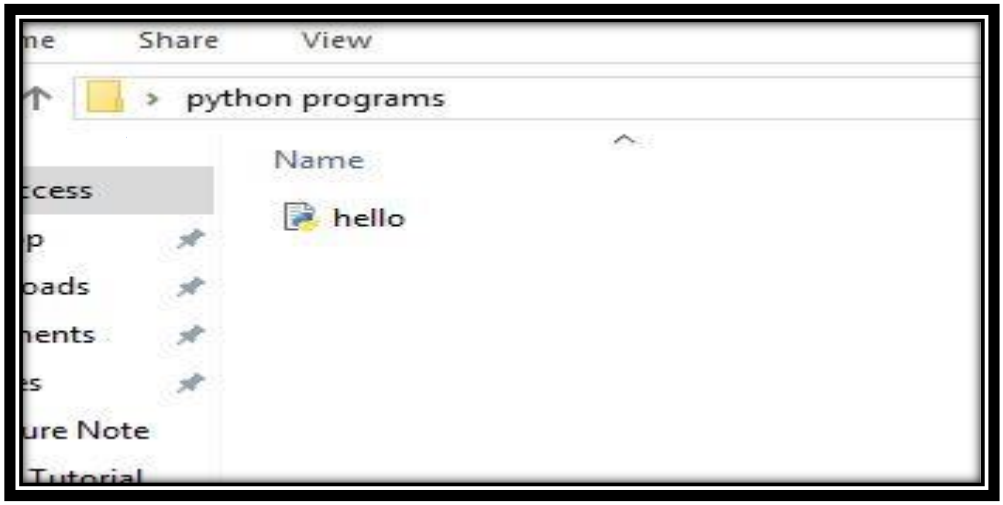


Now create a folder in Desktop. We named it "python programs"

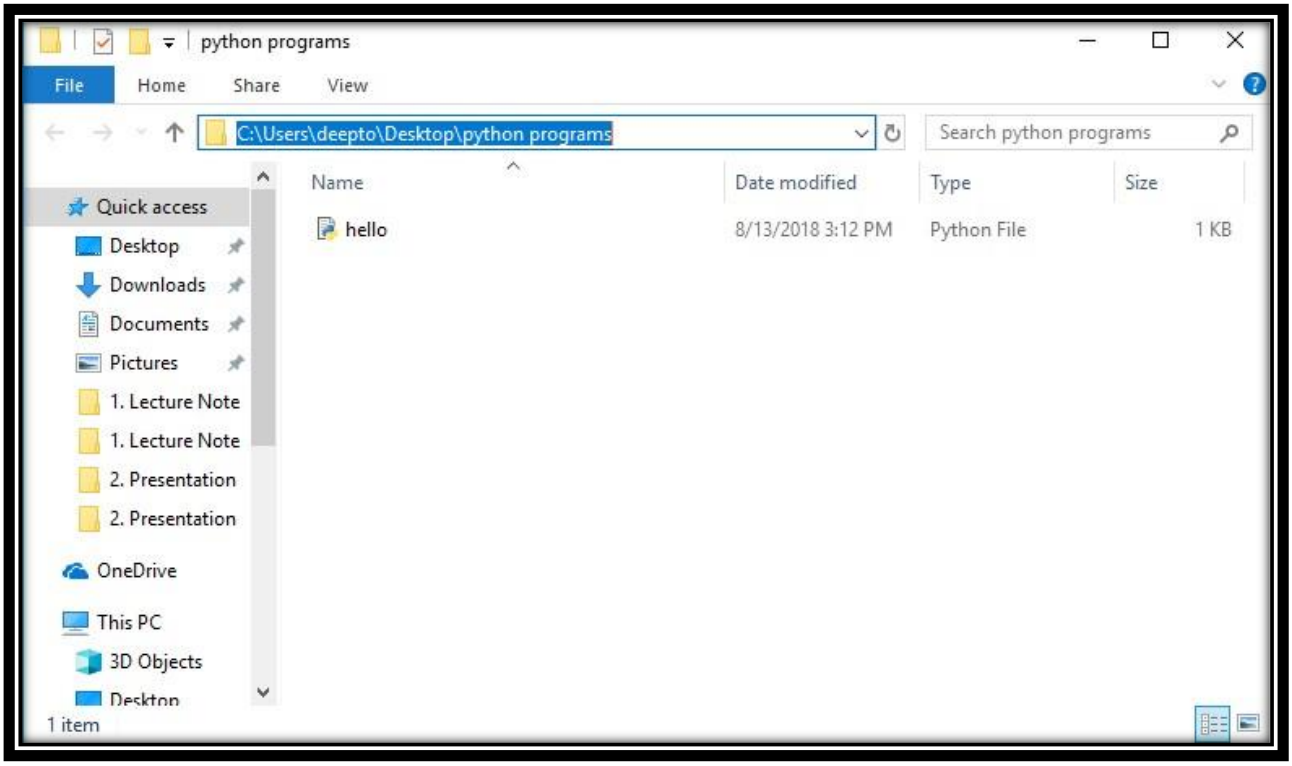
It's time to save our python file on "Python Programs" folder in desktop. Go to file and click on "save as" where you write your code *Print 'hello python'.* Put a name in Files name box. We named it 'hello'. Click on save.

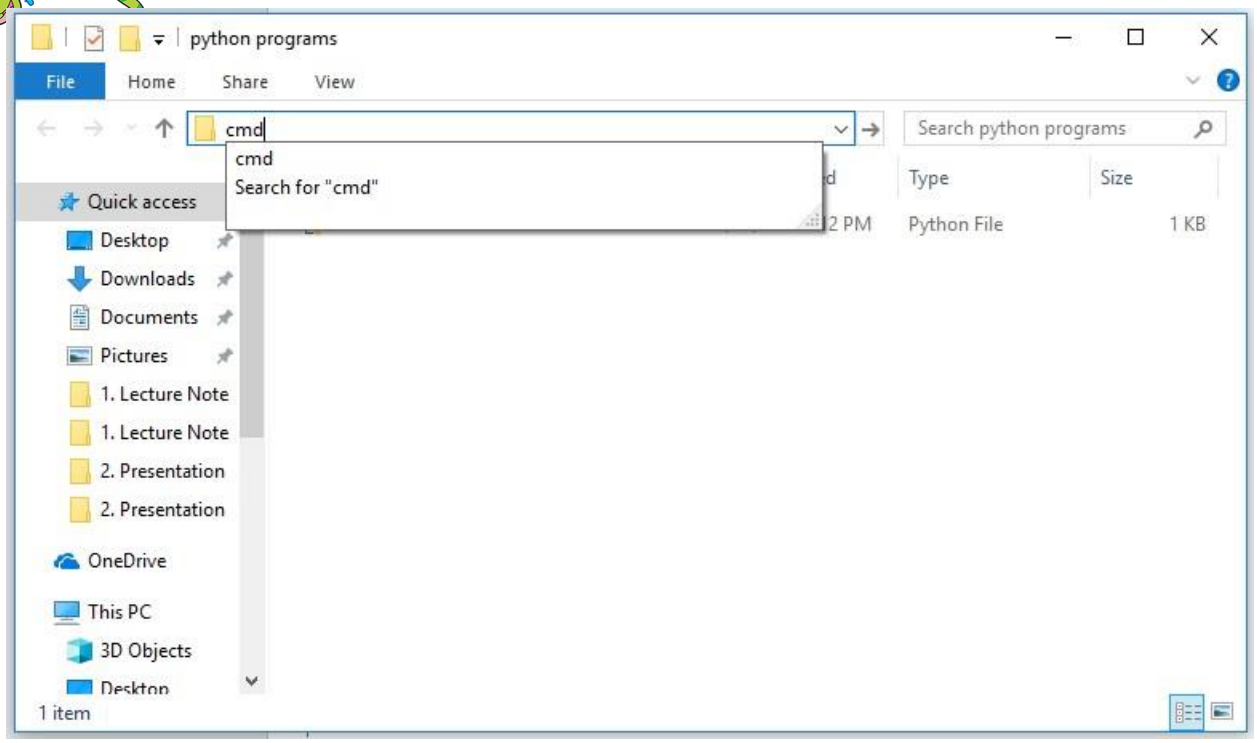


Now you will see your first python file named ‘hello’ in that folder



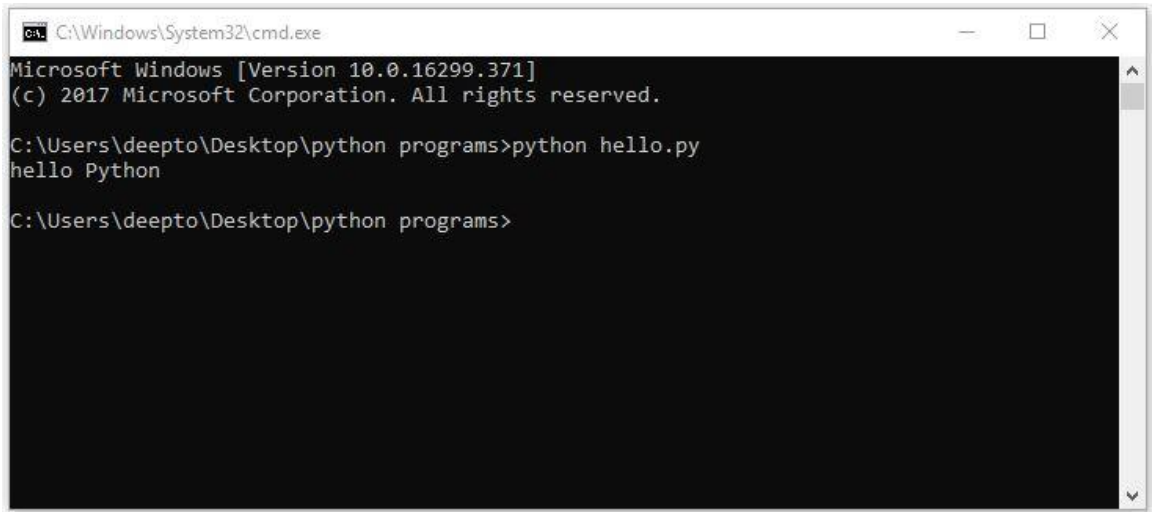
Go to the top of your window click there, type ‘‘cmd’’ and press enter button.

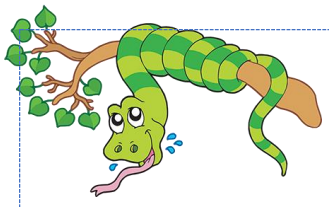




When you press enter a new command prompt window will be opened Type: *python (space) "your_file_name".py* and press enter. As our file name is hello

So we should write→ *python hello.py*

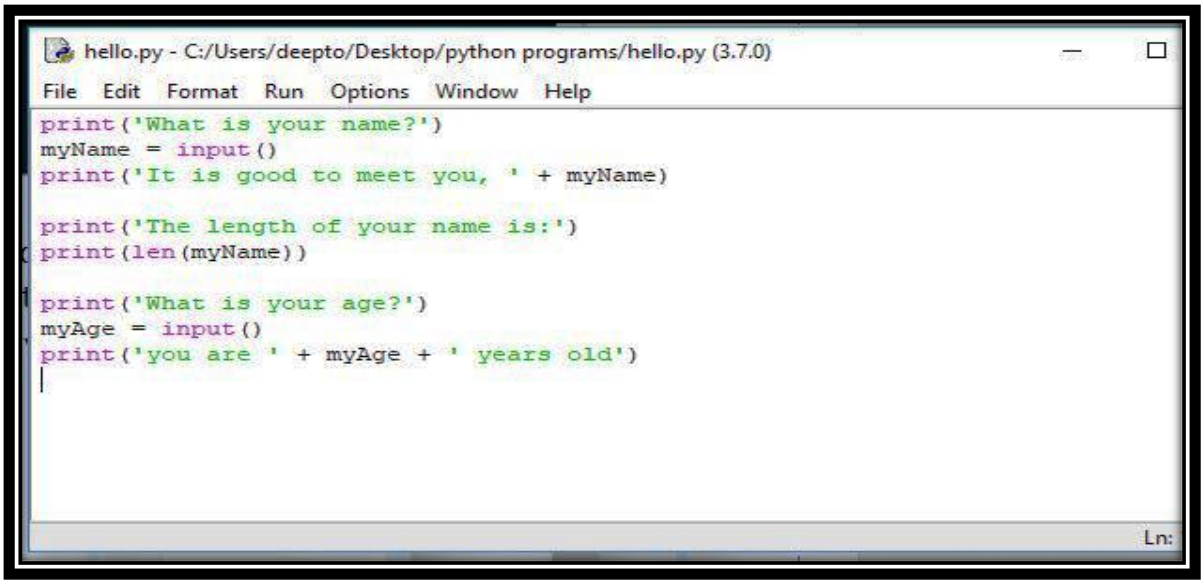




Wow! it works. You commanded your computer to show “hello python” using a function named “print” and your computer has completely listened your command. Now try to command him to show your name.

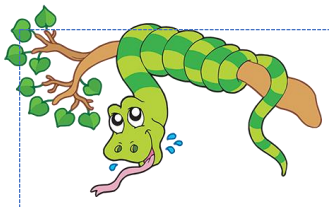
Now write some new code in hello.py file

Press Ctrl+A to mark *Print ‘hello python’* then press Backspace to delete it.



Now Write some new program.

Press Ctrl+s to save it. Try the same process to run the file.



Explanation of code:

1. *print('What is your name?')*
   the program will print “what is your name?”

2. *myName = input ()*
   Here, myName is a variable and we assign an input function in it. So at first program prints “what is your name?” then it will stop for the user input. I typed my name “Deepto” so the variable myName takes it variable as “Deepto”

3. *print('It is good to meet you, ' + myName)*
   I concat two string 'It is good to meet you, ‘ and myName using ‘+’ sign. Note that, ‘it is good to meet you is a string’ but myName is a variable. We should use quote ‘ ’ only for string not for variable. Now as myName means Deepto so after concatenation it will print → It is good to meet you, Deepto

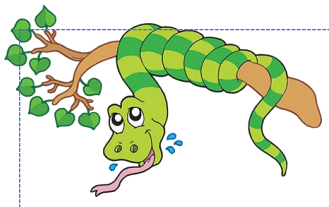
4. *print('The length of your name is:')*
   simply print the following command

5. *print(len(myName))*
   len() is a function which counts the length of a string variable. As “Deepto” has six letters so it will print 6.

6. *print('What is your age?')*
   simply print the following command

7. *myAge = input()*
   asking for user input to store a variable inside myAge

8. *print('you are ' + myAge + ' years old')*
   same as happening in no 3



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.16299.371]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\deepto\Desktop\python programs>python hello.py
What is your name?
Deepto
It is good to meet you, Deepto
The length of your name is:
6
What is your age?
16
you are 16 years old

C:\Users\deepto\Desktop\python programs>
```