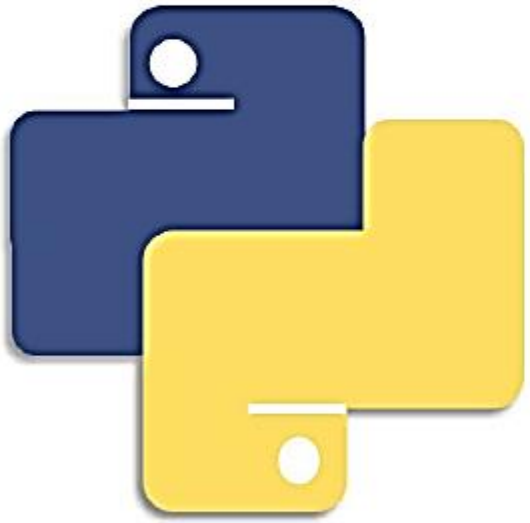
Python

By

Dr. rabih ali sbera

Department of Computer Engineering Techniques-2 nd

2024-2025



Python Programming

Basics of Python

What is python ?

Python is a programming language that lets you work more quickly and integrate your systems more effectively.

(in other words, magic!)



Python Vocab

- ***Program*** – A larger file of code that may contain one or more functions.
- ***Variable*** - names that you can assign values to, allowing you to reuse them later on.

  E.g.: `x = 1` or `msg = "Hi, I'm a message!"`
- ***Comments*** – These are notes ignored by the computer. In Python, comments start with a hash mark (#) and end at the end of the line.

  E.g.:
  ```
  >>> x + y #both variables store user input
  ```
- ***Operators*** – Mathematical symbols, like +, -, *, and /, but also ** (for exponents).

Python Vocab

- ***Keyword*** – Words with meaning/purpose in Python. E.g. "`and`", "`print`", and "`if`".
- ***Function*** – A chunk of code that performs an action. Functions have names and are reusable. Kinds: built-in ones, and "user-defined" ones.
- ***Expression*** – Statements that produce values. Examples include 3 + 5, "Hello world!".
- ***Error*** – When your program has a problem, the command area will provide an error message.

Why to use Python?

The following are the primary factors to use python in day-to-day life:

1. Python is object-oriented

Structure supports such concepts as polymorphism, operation overloading and multiple inheritance.

2. Indentation

Indentation is one of the greatest feature in python

3. It's free (open source)

Downloading python and installing python is free and easy

4. It's Powerful

- ❖ Dynamic typing
- ❖ Built-in types and tools
- ❖ Library utilities
- ❖ Third party utilities (e.g. Numeric, NumPy, sciPy)
- ❖ Automatic memory management

Why to use Python?

5. It’s Portable

Python runs virtually every major platform used today As long as you have a compatible python interpreter installed, python

programs will run in exactly the same manner, irrespective of platform.

6. It’s easy to use and learn

No intermediate compile Python Programs are compiled automatically to an intermediate form called byte code, which the interpreter then reads.

This gives python the development speed of an interpreter without the performance loss inherent in purely interpreted languages.

Structure and syntax are pretty intuitive and easy to grasp.

7. Interpreted Language

Python is processed at runtime by python Interpreter

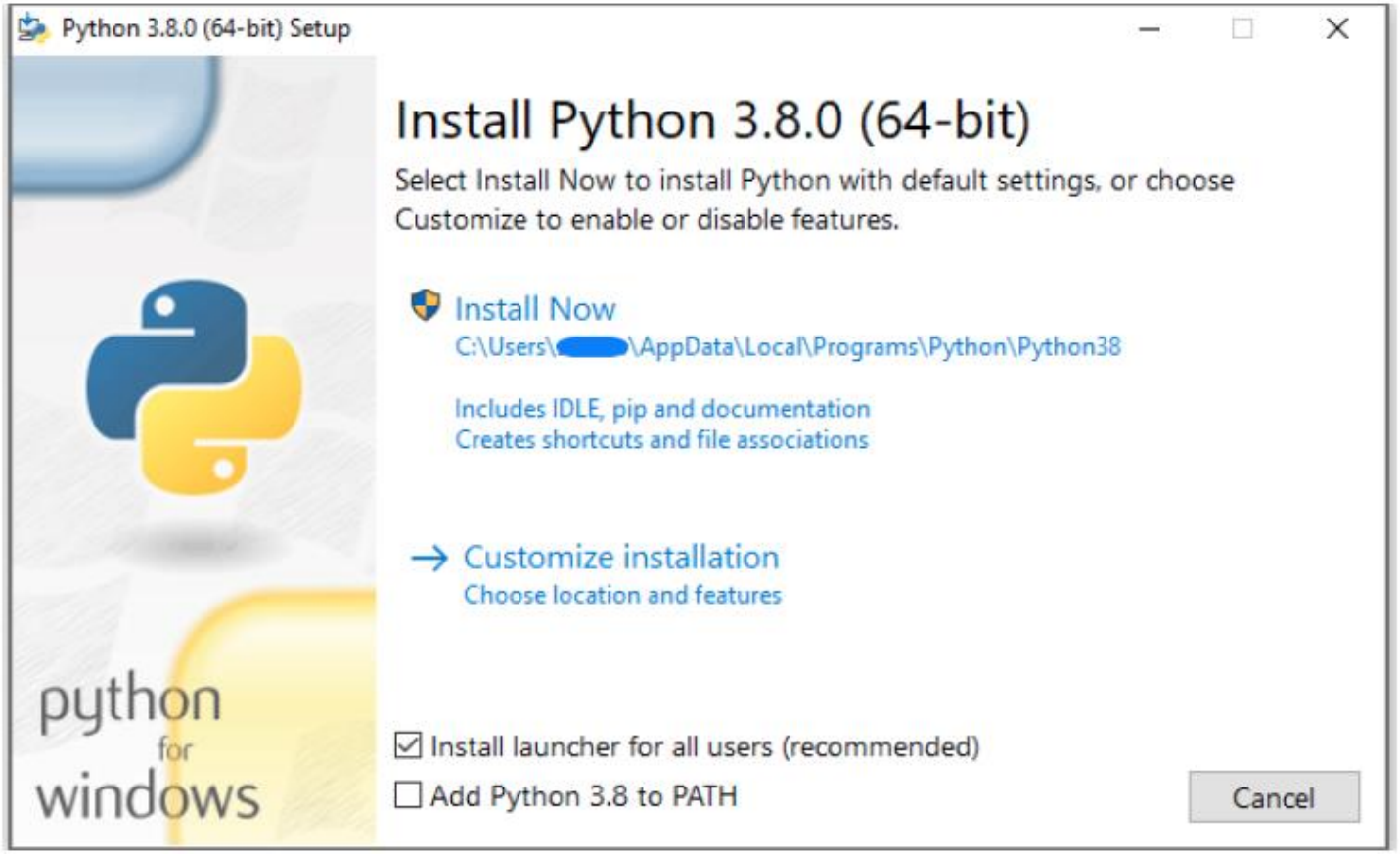
8. Interactive Programming Language

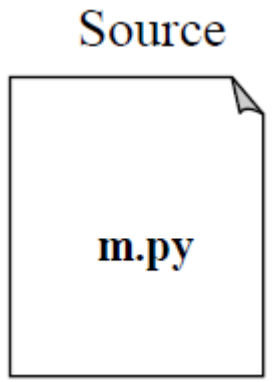
Users can interact with the python interpreter directly for writing the programs

9. Straight forward syntax

The formation of python syntax is simple and straight forward which also makes it popular.

Install Python





Output

E.g. Print:

```
print "Hello world!"
```

will give the output

Hello world!



Similar formatting, different output

- `print "Hello", "world", "!"`
  will output
  Hello world !

- `print "Hello" + "world" + "!"`
  will output
  Helloworld!



Similar formatting, different output

- ```
  print "Hello"
  print "world"
  print "!"
  ```
- will give the output

  Hello
  world
  !

- ```
  print "Hello",
  print "world",
  print "!"
  ```
- will give the output

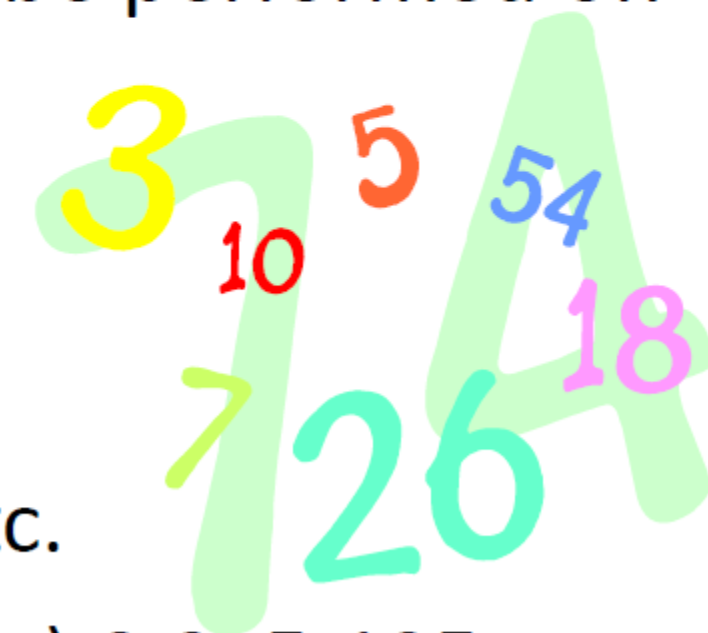
  Hello world !



Data Types

The data type of an object determines the values it can hold, and the operations which can be performed on it.

- Numeric Data



Numeric data comes in 2 main flavors:

– Integers (whole numbers) 2, 5, -7, etc.

– Floating Point Numbers (non-integers) 0.2, 5.125, etc.

Data Types



- Non-numeric Data Types:

  Those include strings (text), lists, dictionaries, etc..

  Basically anything you can not add up using a simple plus sign (+).







Not a String? Not a Problem!

You can also format outputting *variables* you've defined:

- ```
  x = 42
  print“ The value of x is", x, "."“
  ```
- will give the output
  The value of x is 42 .
  ```

Not a String? Not a Problem!

- ```
  x = 42
  print "$" + x
  ```
  causes an error.
- So what do we do?

  ```
  x = 42
  print "$" + str(x)
  ```
  will give the output
  $42
  ```

Defining Variables

Rules for naming variables:

- Have to start with a letter or underscore (_)
- Can contain letters, numbers, and underscores
- Can’t contain spaces, punctuation, etc.
- Can’t be Python keywords
- Are case sensitive

Defining Variables

- For multi-word variable names, two options:
- start capitalizing each word after the first “myCar”.
- separate words with underscores. For instance, a variable for “Ford Focus” could be “my_car”.
- Abbreviating is common for longer words. So, a variable for “average price” could be “avgPrice” or even “avg”.

Variables

- Variables can also be assigned new values that are relative to their old values. For example:

```
total = 10
print "Original total:", total
total = total + 4
print "New total:", total
```

will give the output

Original total: 10

New total: 14

Variables

- Remember: A variable has to have been defined on a previous line before it can be used on the right-hand side of an equation, so:
- ```
  total = total + 4
  print "Total:", total
  ```
  causes an error, since there was no mention of the value of “total” before the line trying to redefine it.
  ```

Numeric Operators

- Python built-in numeric operators:
- + addition
- - subtraction
- * multiplication
- / division
- ** exponentiation
- % remainder (modulo)

Python Arithmetic

- Try writing the following code in your program area and see what it outputs

```
Def main():
    a = 12
    b = 2
    c = 16
    d = 3
    e = 2.5

    print "the value of a is", a
    print (a / b) * 5
    print a + b * d
    print (a + b) * d
    print b ** d
    print c - e
    a = a + b
    print "the value of a is", a
```

Variables

- Variables can hold all kinds of values, including strings, different types of numbers, and user input.
- To assign a string value to a variable, you have to wrap the string in quotes (like usual).

```
firstName = "John"
lastName = "Doe"
mathProblem = "5 + 5"
print lastName, ",", firstName, ";",
mathProblem
```

will give the output

Doe , John ; 5 + 5

Data types

The data stored in memory can be of many types. For example, a student roll number is stored as a numeric value and his or her address is stored as alphanumeric characters.

Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Data types

- **Int:**

Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

- **Float:**

Float, or "floating point number" is a number, positive or negative, containing one or more decimals.

- **Boolean:**

Objects of Boolean type may have one of two values, True or False:

- **String:**

1. Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes.

'hello' is the same as "hello".

```
x = 35
y =1.3
z = "Python"
print(type(x))
print(type(y))
print(type(z))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
```

Data types

- **Suppressing Special Character:**

Specifying a backslash (\) in front of the quote character in a string “escapes” it and causes Python to suppress its usual special meaning. It is then interpreted simply as a literal single quote character:

Escape Sequence	Usual Interpretation of Character(s) After Backslash	“Escaped” Interpretation
`\'`	Terminates string with single quote opening delimiter	Literal single quote (') character
`\"`	Terminates string with double quote opening delimiter	Literal double quote (") character
`\newline`	Terminates input line	Newline is ignored
`\\`	Introduces escape sequence	Literal backslash (\) character

Basics

```
>>> print("mrcet is an autonomous (\') college")

mrcet is an autonomous (') college

>>> print('mrcet is an autonomous (\") college')

mrcet is an autonomous (") college
```

The following is a table of escape sequences which cause Python to suppress the usual special interpretation of a character in a string:

```
>>> print('a\
....b')
a....b
>>> print('a\
b\
c')
```

Output
abc

```
>>> print('a \n b')
a
 b
>>> print("mrcet \n college")
mrcet
 college
```