

**Guido van Rossum**

**Python** is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

**Python** code is translated into intermediate code, which has to be executed by a **virtual machine**, known as the PVM, the Python Virtual Machine. This is a similar approach to the one taken by Java. There is even a way of translating Python programs into Java byte code for the Java Virtual Machine (JVM). This can be achieved with Jython.

**Getting started with Python Language**

**Python 3.x**

**Version Release Date**

* **3.7 -** 2018-06-27
* **3.6 -** 2016-12-23
* **3.5** - 2015-09-13
* **3.4** - 2014-03-17
* **3.3** - 2012-09-29
* **3.2 -** 2011-02-20
* **3.1 -** 2009-06-26
* **3.0 -** 2008-12-03

**Python 2.x**

**Version Release Date**

* **2.7 -** 2010-07-03
* **2.6 -** 2008-10-02
* **2.5 -** 2006-09-19
* **2.4 -** 2004-11-30
* **2.3 -** 2003-07-29
* **2.2 -** 2001-12-21
* **2.1 -** 2001-04-15
* **2.0 -** 2000-10-16

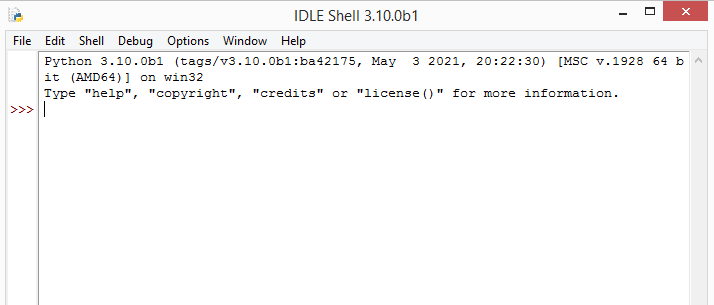
**How to download Python**

1. Head over to <https://python.org/download>
2. Download the latest version of Python
3. Run the .exe file and Install the Python **P.S: Don’t forgot to add to path while downloading**
4. Run CMD and type **python –version** if you get something as given in the following everything is fine.



1. If something went wrong repeat the steps and install again.

**First Program in Python**

1. Type IDLE in windows search bar and hit Enter. Following window will pop up.
2. Type **print('Hello Zakariyan Coding Society!')** and hit Enter.



**Section 1.2: Creating variables and assigning values**

To create a variable in Python, all you need to do is specify the variable name, and then assign a value to it.

**<variable name> = <value>**

Python uses = to assign values to variables. There's no need to declare a variable in advance (or to assign a data type to it), assigning a value to a variable itself declares and initializes the variable with that value. There's no way to declare a variable without assigning it an initial value.

**# Integer**

a = 2

print(a)

# Output: 2

**# Integer**

b = 9223372036854775807

print(b)

# Output: 9223372036854775807

**# Floating point**

pi = 3.14 print(pi)

# Output: 3.14

**# String**

c = 'A'

print(c)

# Output: A

**# String**

name = 'John Doe'

print(name)

# Output: John Doe

**# Boolean**

q = True

print(q)

# Output: True

**# Empty value or null data type**

x = None

print(x)

# Output: None

**# complex: Complex numbers**

a = 2 + 1j

b = 100 + 10j

**Multiple assigning at one time**

a, b, c = 1, 2, 3

print(a, b, c)

# Output: 1 2 3

a = b = c = 1

print(a, b, c)

**@ Static vs Dynamic Datatype**

**How to check the Type of variables**

a = 2

print(type(a))

# Output: <type 'int'>

b = 9223372036854775807

print(type(b))

# Output: <type 'int'>

pi = 3.14

print(type(pi))

# Output: <type 'float'>

c = 'A'

print(type(c))

# Output: <type 'str'>

name = 'John Doe'

print(type(name))

# Output: <type 'str'>

q = True

print(type(q))

# Output: <type 'bool'>

**Converting between datatypes**

You can perform explicit datatype conversion.

**For example,**

'123' is of str type and it can be converted to integer using int function.

a = '123' b = int(a)

Converting from a ﬂoat string such as '123.456' can be done using float function.

a = '123.456'

b = float(a)

c = int(a) # ValueError: invalid literal for int() with base 10: '123.456'

d = int(b) # 123

You can also convert sequence or collection types

a = 'hello'

list(a) # ['h', 'e', 'l', 'l', 'o']

set(a) # {'o', 'e', 'l', 'h'}

tuple(a) # ('h', 'e', 'l', 'l', 'o')

**Mutable and Immutable Data Types**

An object is called mutable if it can be changed.

For example, when you pass a list to some function, the list can be changed:

**Examples of immutable Data Types:**

* int, long, float, complex
* str
* bytes
* tuple
* frozenset

**Examples of mutable Data Types:**

* bytearray
* list
* set
* dict

**Section 1.3: Block Indentation**

Python uses indentation to deﬁne control and loop constructs. This contributes to Python's readability, however, it requires the programmer to pay close attention to the use of whitespace. Thus, editor miscalibration could result in code that behaves in unexpected ways.

Python uses the colon symbol (:) and indentation for showing where blocks of code begin and end (If you come from another language, do not confuse this with somehow being related to the ternary operator). That is, blocks in Python, such as functions, loops, if clauses and other constructs, have no ending identiﬁers. All blocks start with a colon and then contain the indented lines below it.

**For example:**

def my\_function(): # This is a function definition. Note the colon (:)

a = 2 # This line belongs to the function because it's indented

return a # This line also belongs to the same function

print(my\_function()) # This line is OUTSIDE the function block

or

if a > b: # If block starts here

print(a) # This is part of the if block

else: # else must be at the same level as if

print(b) # This line is part of the else block

Blocks that contain exactly one single-line statement may be put on the same line, though this form is generally not considered good style:

if a > b:

print(a)

else:

print(b)

if x > y: y = x

print(y) # IndentationError: unexpected indent

if x > y: while y != z: y -= 1 # SyntaxError: invalid syntax

def will\_be\_implemented\_later():

pass