UNIT NO. 01 [Marks: 08]

Introduction & Syntax of Python Program

Python is a simple, general purpose, high level, and object-oriented programming language.

Python is an interpreted scripting language also. ***Guido Van Rossum*** is known as the founder of Python programming.

What is Python

1) Python is a general purpose, object-orinted, dynamic, high-level, and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

2) Python is *easy to learn* yet powerful and versatile scripting language, which makes it attractive for Application Development.

3) Python's syntax and *dynamic typing* with its interpreted nature make it an ideal language for scripting and rapid application development.

4) Python supports *multiple programming pattern*, including object-oriented, imperative, and functional or procedural programming styles.

5) Python is not intended to work in a particular area, such as web programming. That is why it is known as *multipurpose* programming language because it can be used with web, enterprise, 3D CAD, etc.

6) We don't need to use data types to declare variable because it is *dynamically typed* so we can write a=10 to assign an integer value in an integer variable.

7) Python makes the development and debugging *fast* because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

Python History:

Python was invented by **Guido van Rossum** in 1991 at CWI in Netherland. The idea of Python programming language has taken from the ABC programming language or we can say that ABC is a predecessor of Python language.

There is also a fact behind the choosing name Python. Guido van Rossum was a fan of the popular BBC comedy show of that time, **"Monty Python's Flying Circus"**. So he decided to pick the name **Python** for his newly created programming language.

Python has the vast community across the world and releases its version within the short period.

- Python laid its foundation in the late 1980s.
- The implementation of Python was started in December 1989 by **Guido Van Rossum** at CWI in Netherland.
- In February 1991, **Guido Van Rossum** published the code (labeled version 0.9.0) to alt.sources.
- In 1994, Python 1.0 was released with new features like lambda, map, filter, and reduce.
- Python 2.0 added new features such as list comprehensions, garbage collection systems.
- On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify the fundamental flaw of the language.
- *ABC programming language* is said to be the predecessor of Python language, which was capable of Exception Handling and interfacing with the Amoeba Operating System.
- The following programming languages influence Python:
    - ABC language.
    - Modula-3

Why the Name Python?

There is a fact behind choosing the name Python

Guido van Rossum was reading the script of a popular BBC comedy series "**Monty Python's Flying Circus**". It was late on-air 1970s.

Van Rossum wanted to select a name which unique, sort, and little-bit mysterious. So he decided to select naming Python after the **"Monty Python's Flying Circus"** for their newly created programming language.

The comedy series was creative and well random. It talks about everything. Thus it is slow and unpredictable, which made it very interesting.

Python is also versatile and widely used in every technical field, such as
Machine Learning,

Why learn Python?

Python provides many useful features to the programmer. These features make it most popular and widely used language. We have listed below few-essential feature of Python.

- Easy to use and Learn
- Expressive Language
- Interpreted Language
- Object-Oriented Language
- Open Source Language
- Extensible
- Learn Standard Library
- GUI Programming Support
- Integrated
- Embeddable
- Dynamic Memory Allocation
- Wide Range of Libraries and Frameworks
- Data Analysis and Processing
- Artificial Intelligence
- Games
- Hardware/Sensor/Robots
- Desktop Applications

Where is Python used?

Python is a general-purpose, popular programming language and it is used in almost every technical field. The various areas of Python use are given below.

- Data Science
- Date Mining
- Desktop Applications
- Console-based Applications
- Mobile Applications
- Software Development
- Artificial Intelligence

- Web Applications
- Enterprise Applications
- 3D CAD Applications
- Machine Learning
- Computer Vision or Image Processing Applications.
- Speech Recognitions
- Scientific Computing
- Robotics
- **Internet of Things (IoT)**
- **Gaming**
- Mobile Apps
- Data Analysis and Preprocessing

Python Basic Syntax:

There is no use of curly braces or semicolon in Python programming language. It is English-like language. But Python uses the indentation to define a block of code. Indentation is nothing but adding whitespace before the statement when it is needed. **For example -**

```
def func():
    statement 1
    statement 2
    …………………
    …………………
    statement N
```

In the above example, the statements that are same level to right belong to the function. Generally, we can use four whitespaces to define indentation.

Python Popular Frameworks and Libraries

Python has wide range of libraries and frameworks widely used in various fields such as machine learning, artificial intelligence, web applications, etc. We define some popular frameworks and libraries of Python as follows.

- **Web development (Server-side) -** Django Flask, Pyramid, CherryPy
- **GUIs based applications -** Tk, PyGTK, PyQt, PyJs, etc.
- **Machine Learning -** TensorFlow, PyTorch, **Scikit-learn**, Matplotlib, Scipy, etc.
- **Mathematics -** Numpy, Pandas, etc.

Python print() Function

The **print()** function displays the given object to the standard output device (screen) or to the text stream file.

Unlike the other programming languages, Python **print()** function is most unique and versatile function.

The syntax of **print()** function is given below.

1. print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)

Let's explain its parameters one by one.

- **objects -** An object is nothing but a statement that to be printed. The * sign represents that there can be multiple statements.
- **sep -** The **sep** parameter separates the print values. Default values is ' '.
- **end -** The **end** is printed at last in the statement.
- **file -** It must be an object with a write(string) method.
- **flush -** The stream or file is forcibly flushed if it is true. By default, its value is false.

Explore Libraries and Frameworks

Python consists of vast libraries and various frameworks. After getting familiar with Python's basic concepts, the next step is to explore the Python libraries. Libraries are essential to work with the domain specific projects. In the following section, we describe the brief introduction of the main libraries.

1) TensorFlow

- It is an artificial intelligence library which allows us to create large scale AI based projects.

2) Django

- It is an open source framework that allows us to develop web applications. It is easy, flexible, and simple to manage.

3) Flask

- It is also an open source web framework. It is used to develop lightweight web applications.

4) Pandas

- It is a Python library which is used to perform scientific computations.

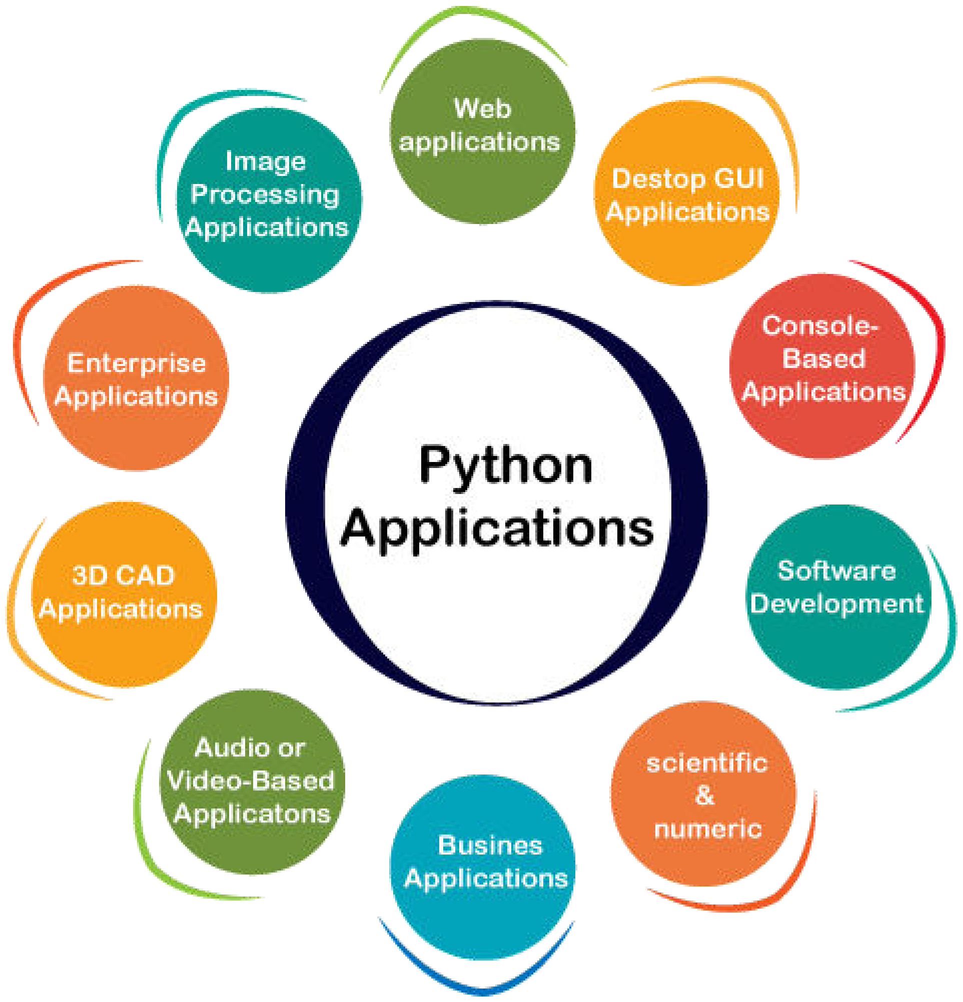
5) Keras

- It is an open source library, which is used to work around the neural network.

Python Applications

Python is known for its general-purpose nature that makes it applicable in almost every domain of software development. Python makes its presence in every emerging field. It is the fastest-growing programming language and can develop any application.

Here, we are specifying application areas where Python can be applied.



1) Web Applications

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautifulSoup, Feedparser, etc. One of Python web-framework named Django is used on **Instagram**. Python provides many useful frameworks, and these are given below:

1) Django and Pyramid framework(Use for heavy applications)
2) Flask and Bottle (Micro-framework)
3) Plone and Django CMS (Advance Content management)

2) Desktop GUI Applications

The GUI stands for the Graphical User Interface, which provides a smooth interaction to any application. Python provides a **Tk GUI library** to develop a user interface. Some popular GUI libraries are given below.

1) Tkinter or Tk
2) wxWidgetM
3) Kivy (used for writing multitouch applications)
4) PyQt or Pyside

3) Console-based Application

- Console-based applications run from the command-line or shell. These applications are computer program which are used commands to execute. This kind of application was more popular in the old generation of computers. Python can develop this kind of application very effectively.

- Python provides many free library or module which helps to build the command-line apps. The necessary **IO** libraries are used to read and write. It helps to parse argument and create console help text out-of-the-box. There are also advance libraries that can develop independent console apps.

4) Software Development

Python is useful for the software development process. It works as a support language and can be used to build control and management, testing, etc.

1) SCons is used to build control.
2) Buildbot and **Apache** Gumps are used for automated continuous compilation and testing.
3) Round or **Trac** for bug tracking and project management.

5) Scientific and Numeric

This is the era of Artificial intelligence where the machine can perform the task the same as the human. Python language is the most suitable language for Artificial intelligence or machine learning. It consists of many scientific and mathematical libraries, which makes easy to solve complex calculations.

Implementing machine learning algorithms require complex mathematical calculation. Python has many libraries for scientific and numeric such as Numpy, Pandas, Scipy, Scikit-learn, etc. Few popular frameworks of machine libraries are given below.

1) SciPy
2) Scikit-learn
3) NumPy
4) Pandas
5) Matplotlib

6) Business Applications

Business Applications differ from standard applications. E-commerce and ERP are an example of a business application. This kind of application requires extensively, scalability and readability, and Python provides all these features.

7) Audio or Video-based Applications

Python is flexible to perform multiple tasks and can be used to create multimedia applications. Some multimedia applications which are made by using Python are **TimPlayer, cplay,** etc. The few multimedia libraries are given below.

1) Gstreamer
2) Pyglet
3) QT Phonon

8) 3D CAD Applications

The CAD (Computer-aided design) is used to design engineering related architecture. It is used to develop the 3D representation of a part of a system. Python can create a 3D CAD application by using the following functionalities.

1) vFandango (Popular)
2) CAMVOX
3) HeeksCNC
4) AnyCAD
5) RCAM

9) Enterprise Applications

Python can be used to create applications that can be used within an Enterprise or an Organization. Some real-time applications are OpenERP, Tryton, Picalo, etc.

10) Image Processing Application

Python contains many libraries that are used to work with the image. The image can be manipulated according to our requirements. Some libraries of image processing are given below.

1) OpenCV
2) Pillow
3) SimpleITK

In this topic, we have described all types of applications where Python plays an essential role in the development of these applications.

Python Features :

Python provides many useful features which make it popular and valuable from the other programming languages. It supports object-oriented programming, procedural programming approaches and provides dynamic memory allocation. We have listed below a few essential features.

1) Easy to Learn and Use:

Python is easy to learn as compared to other programming languages. Its syntax is straightforward and much the same as the English language.

There is no use of the semicolon or curly-bracket, the indentation defines the code block. It is the recommended programming language for beginners.

2) Expressive Language:

Python can perform complex tasks using a few lines of code. A simple example, the hello world program you simply type **print("Hello World")**. It will take only one line to execute, while Java or C takes multiple lines.

3) Interpreted Language:(imp)

Python is an interpreted language; it means the Python program is executed one line at a time. The advantage of being interpreted language, it makes debugging easy and portable.

4) Platform independent / Cross-platform Language:(imp)

Python can run equally on different platforms such as Windows, Linux, UNIX, and Macintosh, etc. So, we can say that Python is a portable language.

It enables programmers to develop the software for several competing platforms by writing a program only once.

5) Free and Open Source:

Python is freely available for everyone. It has a large community across the world that is dedicatedly working towards make new python modules and functions.

Anyone can contribute to the Python community. The open-source means, "Anyone can download its source code without paying any penny."

6) Object-Oriented Language: (imp)

Python supports object-oriented language and concepts of classes and objects come into existence.

It supports inheritance, polymorphism, and encapsulation, etc. The object-oriented procedure helps to programmer to write reusable code and develop applications in less code.

7) Extensible:

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our Python code.

It converts the program into byte code, and any platform can use that byte code.

8) Large Standard Library:

It provides a vast range of libraries for the various fields such as machine learning, web developer, and also for the scripting.

There are various machine learning libraries, such as Tensor flow, Pandas, Numpy, Keras, and Pytorch, etc. Django, flask, pyramids are the popular framework for Python web development.

9) GUI Programming Support:

Graphical User Interface is used for the developing Desktop application. PyQT5, Tkinter, Kivy are the libraries which are used for developing the web application.

10) Integrated

It can be easily integrated with languages like C, C++, and JAVA, etc. Python runs code line by line like C,C++ Java. It makes easy to debug the code.

The code of the other programming language can use in the Python source code. We can use Python source code in another programming language as well. It can embed other language into our code.

11) Dynamic Memory Allocation: (imp)

In Python, we don't need to specify the data-type of the variable.

When we assign some value to the variable, it automatically allocates the memory to the variable at run time.

Suppose we are assigned integer value 15 to **x,** then we don't need to write **int x = 15.** Just write x = 15.

12) Interactive: (imp)

Python is interactive. When a Python statement is entered, and is followed by the Return key, if appropriate, the result will be printed on the screen, immediately, in the next line.

This is particularly advantageous in the debugging process. In interactive mode of operation, Python is used in a similar way as the Unix command line or the terminal.

Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Python Building Block:

1) Identifiers :

Variables are the example of identifiers. An Identifier is used to identify the literals used in the program. The rules to name an identifier are given below.

1) The first character of the variable must be an alphabet or underscore (_).
2) All the characters except the first character may be an alphabet of lower-case(a-z), upper-case (A-Z), underscore, or digit (0-9).

3) Identifier name must not contain any white-space, or special character (!,@, #, %, ^, &, *).

4) Identifier name must not be similar to any keyword defined in the language.

5) Identifier names are case sensitive; for example, my name, and MyName is not the same.

6) Examples of valid identifiers: a123, _n, n_9, etc.

7) Examples of invalid identifiers: 1a, n%4, n 9, etc.

2) Python Variables:

1) Variable is a name that is used to refer to memory location. Python variable is also known as an identifier and used to hold value.

2) In Python, we don't need to specify the type of variable because Python is a infer language and smart enough to get variable type.

3) Variable names can be a group of both the letters and digits, but they have to begin with a letter or an underscore.

4) It is recommended to use lowercase letters for the variable name. Rahul and rahul both are two different variables.

3) Python indentation:

1) Python indentation uses to define the block of the code. The other programming languages such as C, C++, and Java use curly braces {}, whereas Python uses an indentation. Whitespaces are used as indentation in Python.

2) Indentation uses at the beginning of the code and ends with the unintended line. That same line indentation defines the block of the code (body of a function, loop, etc.)

3) Generally, four whitespaces are used as the indentation. The amount of indentation depends on user, but it must be consistent throughout that block.

4) Consider the following example.

```
dn = int(input("Enter the number:"))
if(n%2 == 0):
    print("Even Number")
else:
    print("Odd Number")


 print("Task Complete")
```

Output: Enter the number: 10

```
Even Number
```

The above code, **if** and **else** are two separate code blocks. Both code blocks are indented four spaces. The print("Task Complete") statement is not indented four whitespaces and it is out of the **if-else** block.

If the indentation is not used properly, then that will result in **IndentationError**.

4) Python Keywords:

Python Keywords are special reserved words that convey a special meaning to the compiler/interpreter. Each keyword has a special meaning and a specific operation. These keywords can't be used as a variable. Following is the List of Python Keywords.

True	False	None	And	as
Asset	Def	Class	Continue	break
Else	Finally	Elif	Del	except
Global	For	If	from	import
Raise	Try	Or	return	pass
Nonlocal	In	Not	is	lambda

5) Python Comments

1) Python Comment is an essential tool for the programmers. Comments are generally used to explain the code. We can easily understand the code if it has a proper explanation.

2) A good programmer must use the comments because in the future anyone wants to modify the code as well as implement the new module; then, it can be done easily.

3) In the other programming language such as C++, It provides the // for single-lined comment and /*.... */ for multiple-lined comment, but Python provides the single-lined Python comment. To apply the comment in the code we use the hash(#) at the beginning of the statement or code.

Let's understand the following example.

```
# This is the print statement
    print("Hello Python")
```

Here we have written comment over the print statement using the hash(#). It will not affect our print statement.

Multiline Python Comment:

We must use the hash(#) at the beginning of every line of code to apply the multiline Python comment. Consider the following example.

```
# First line of the comment
# Second line of the comment
# Third line of the comment
```

Example:

```
a = 5                          # Variable a holds value 5
b = 10                         # Variable b holds value 10
c = a+b                        # Variable c holds sum of a and b
print("The sum is:", c)        # Print the result
```

The above code is very readable even the absolute beginners can under that what is happening in each line of the code. This is the advantage of using comments in code.

How to Install Python (Environment Set-up)

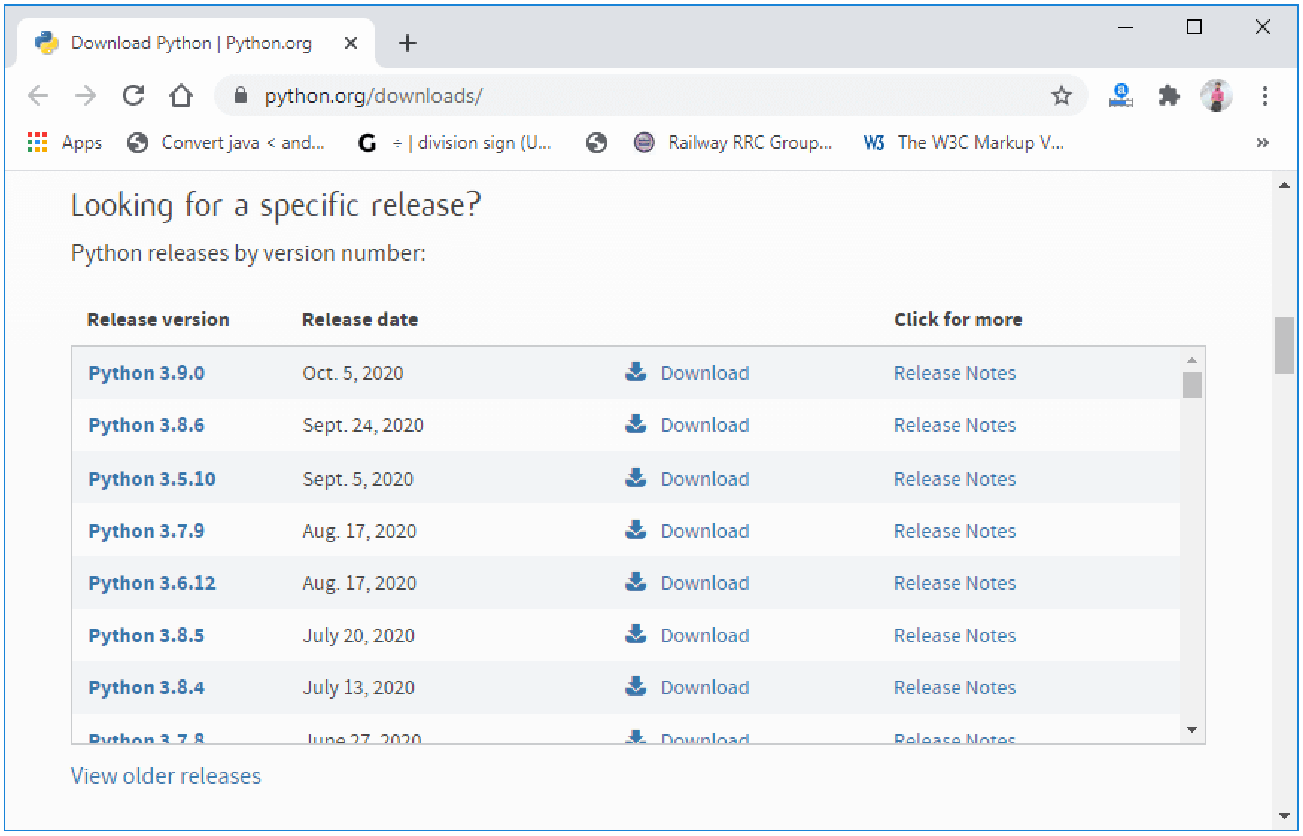
In order to become Python developer, the first step is to learn how to install or update Python on a local machine or computer. we will discuss the installation of Python on various operating systems.

Installation on Windows

Visit the link *https://www.python.org/downloads/* to download the latest release of Python. In this process, we will install Python 3.8.6 on our Windows operating system. When we click on the above link, it will bring us the following page.

Step - 1: Select the Python's version to download.

Click on the download button.


Looking for a specific release?

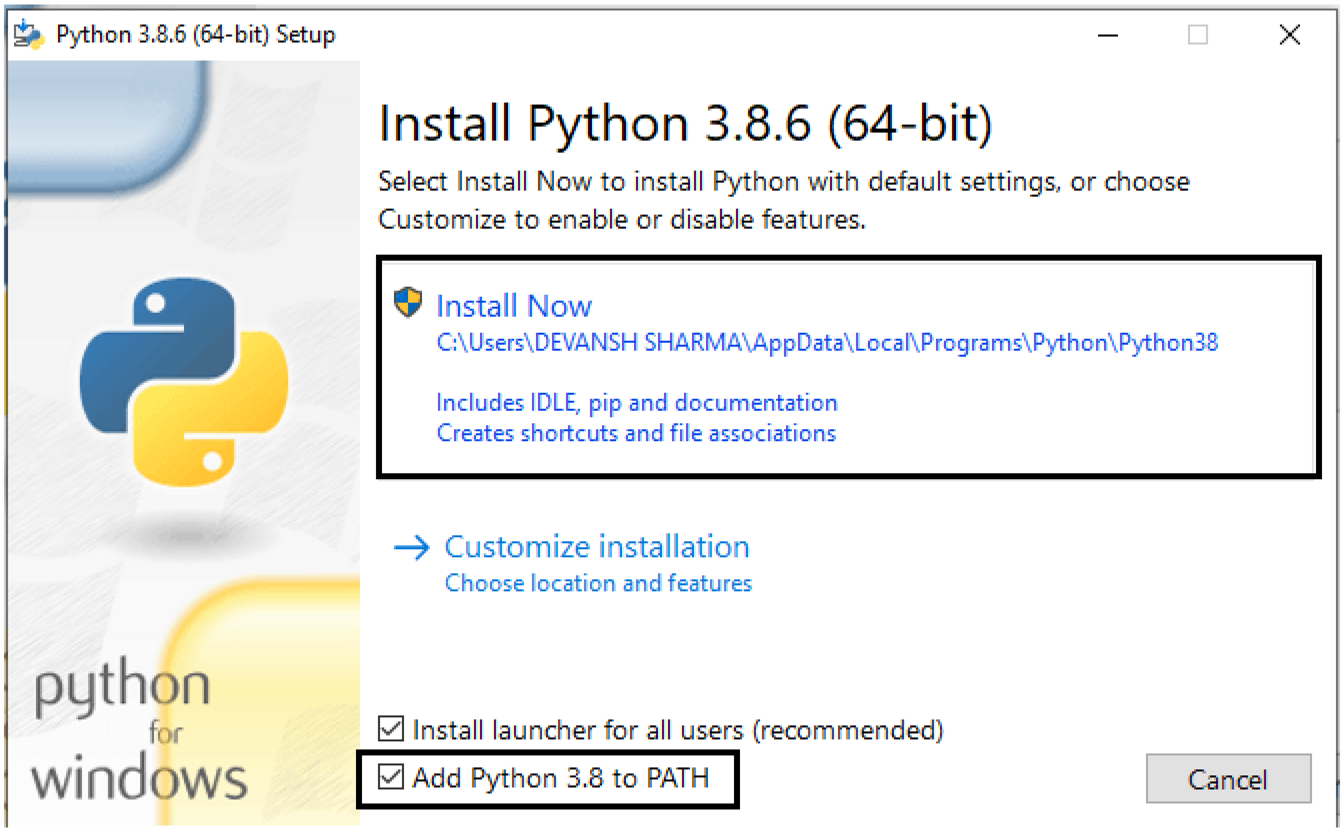
Python releases by version number:

Release version	Release date		Click for more
Python 3.9.0	Oct. 5, 2020	Download	Release Notes
Python 3.8.6	Sept. 24, 2020	Download	Release Notes
Python 3.5.10	Sept. 5, 2020	Download	Release Notes
Python 3.7.9	Aug. 17, 2020	Download	Release Notes
Python 3.6.12	Aug. 17, 2020	Download	Release Notes
Python 3.8.5	July 20, 2020	Download	Release Notes
Python 3.8.4	July 13, 2020	Download	Release Notes
Python 3.7.8	June 27, 2020	Download	Release Notes

View older releases

Step - 2: Click on the Install Now

Double-click the executable file, which is downloaded; the following window will open. Select Customize installation and proceed. Click on the Add Path check box, it will set the Python path automatically.


Python 3.8.6 (64-bit) Setup

Install Python 3.8.6 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\DEVANSH SHARMA\AppData\Local\Programs\Python\Python38

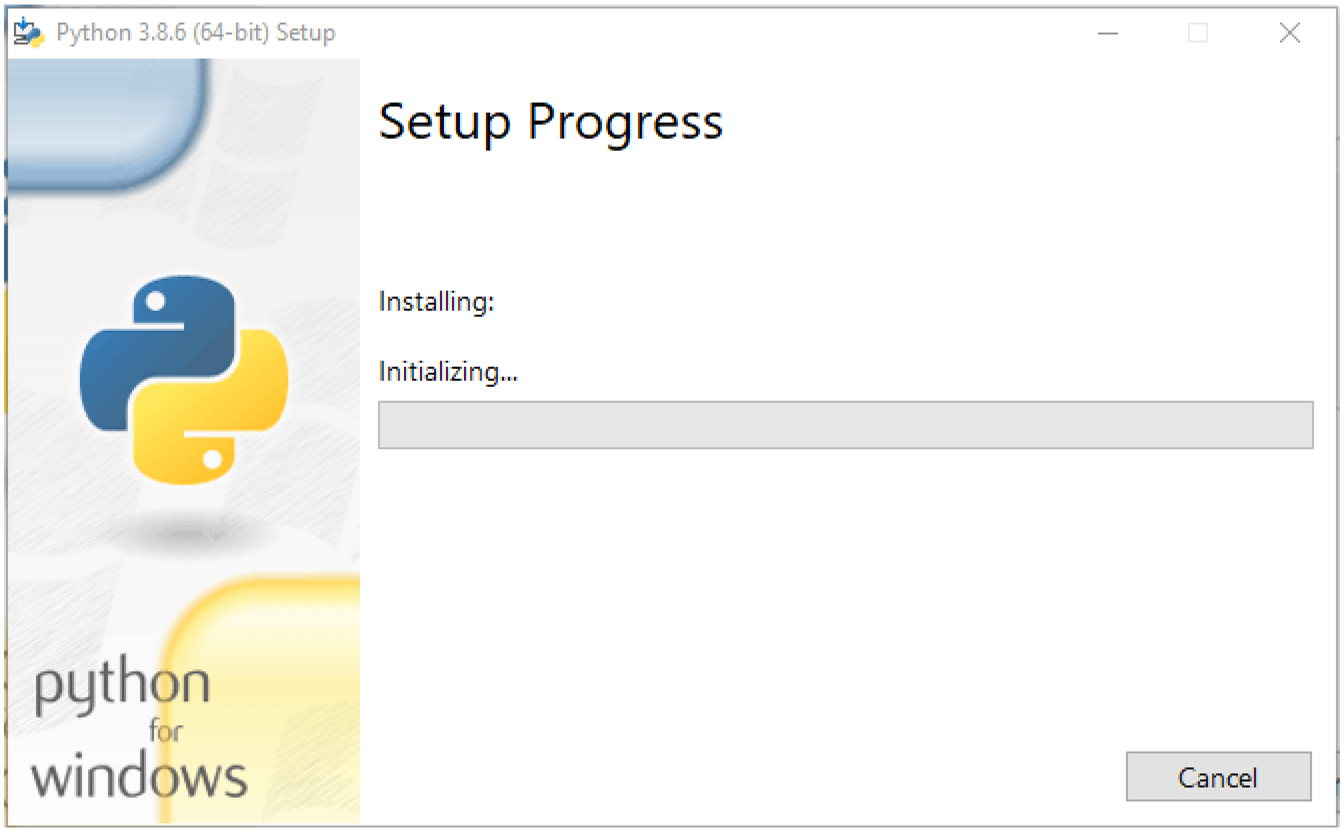
Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.8 to PATH

We can also click on the customize installation to choose desired location and features. Other important thing is install launcher for the all user must be checked.

Step - 3 Installation in Process

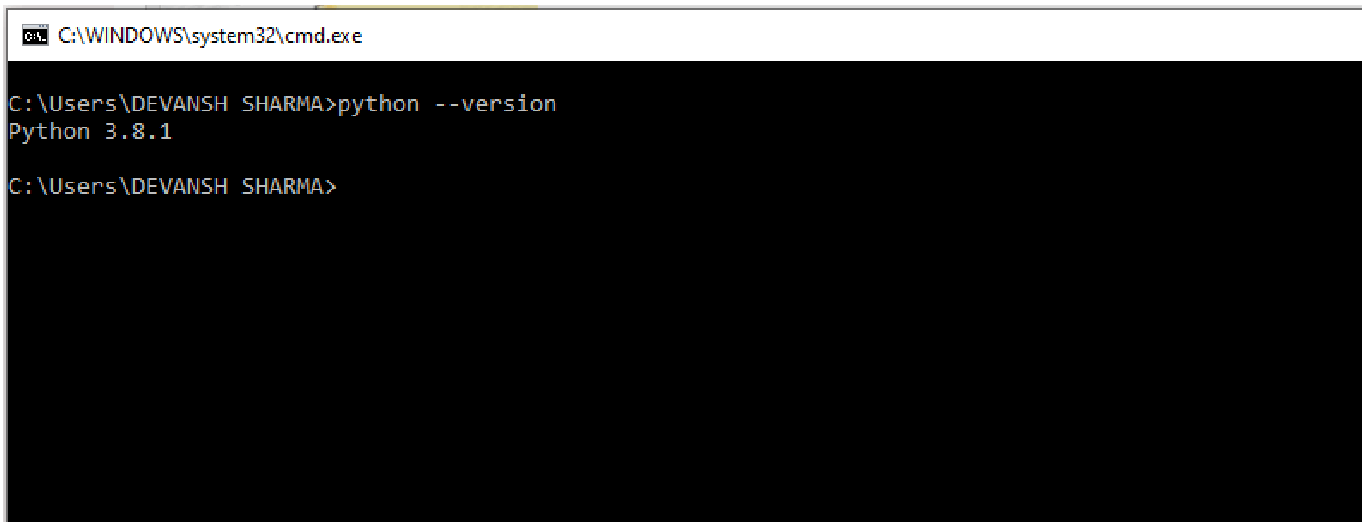


Setup Progress

Installing:

Initializing...

Now, try to run python on the command prompt. Type the command python -version in case of python3.



```
C:\Users\DEVANSH SHARMA>python --version
Python 3.8.1

C:\Users\DEVANSH SHARMA>
```

We are ready to work with the Python.

First Python Program

In this Section, we will discuss the basic syntax of Python, we will run a simple program to print **Hello World** on the console.

Python provides us the two ways to run a program:

- Using Interactive interpreter prompt
- Using a script file

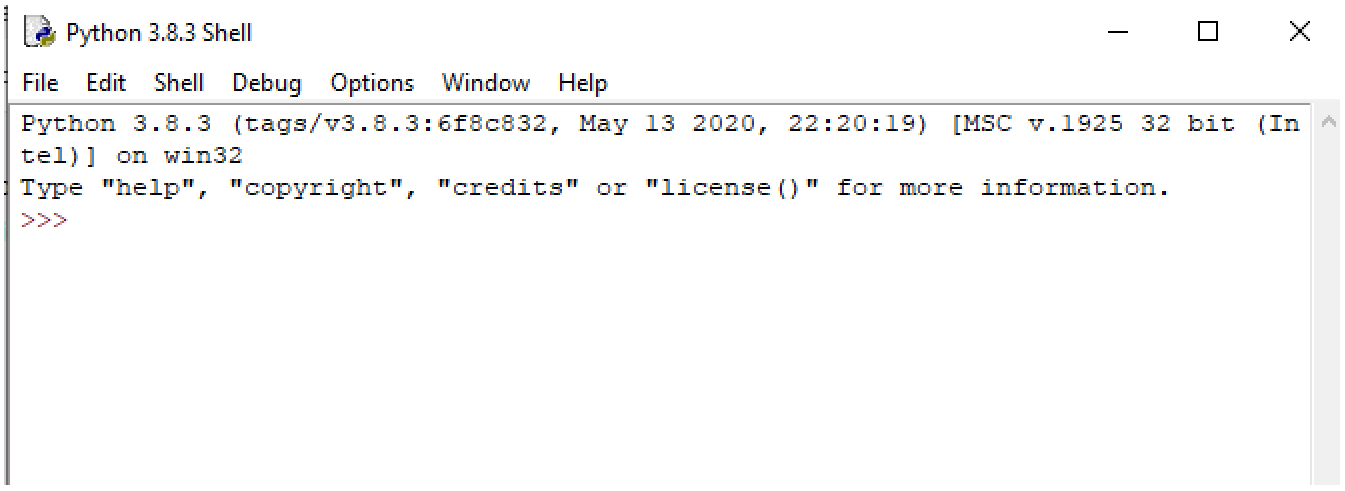
Let's discuss each one of them in detail.

Interactive interpreter prompt

Python provides us the feature to execute the Python statement one by one at the interactive prompt. It is preferable in the case where we are concerned about the output of each line of our Python programLearn Python Programming

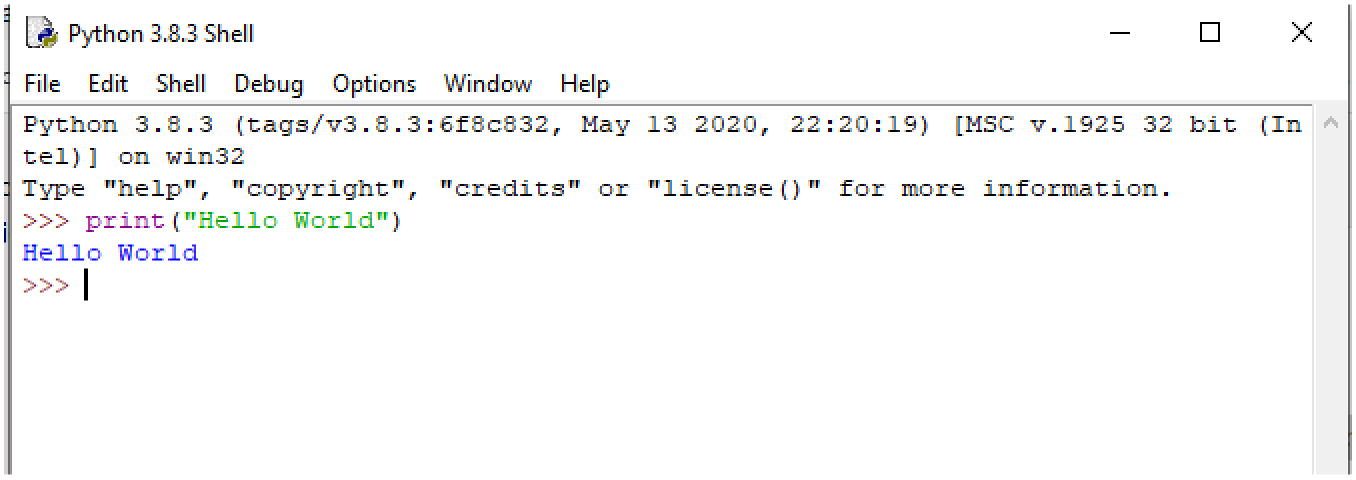
To open the interactive mode, open the terminal (or command prompt) and type python (python3 in case if you have Python2 and Python3 both installed on your system).

It will open the following prompt where we can execute the Python statement and check their impact on the console.



```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

After writing the print statement, press the **Enter** key.



```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>> 
```

Here, we get the message **"Hello World !"** printed on the console.

Python Data Types:

1) Variables can hold values, and every value has a data-type. Python is a dynamically typed language; hence we do not need to define the type of the variable while declaring it. The interpreter implicitly binds the value with its type.

Eg. a = 5

2) The variable **a** holds integer value five and we did not define its type. Python interpreter will automatically interpret variables **a** as an integer type.

3) Python enables us to check the type of the variable used in the program. Python provides us the **type()** function, which returns the type of the variable passed.

Consider the following example to define the values of different data types and checking its type..4M SQL CREATE TABLE

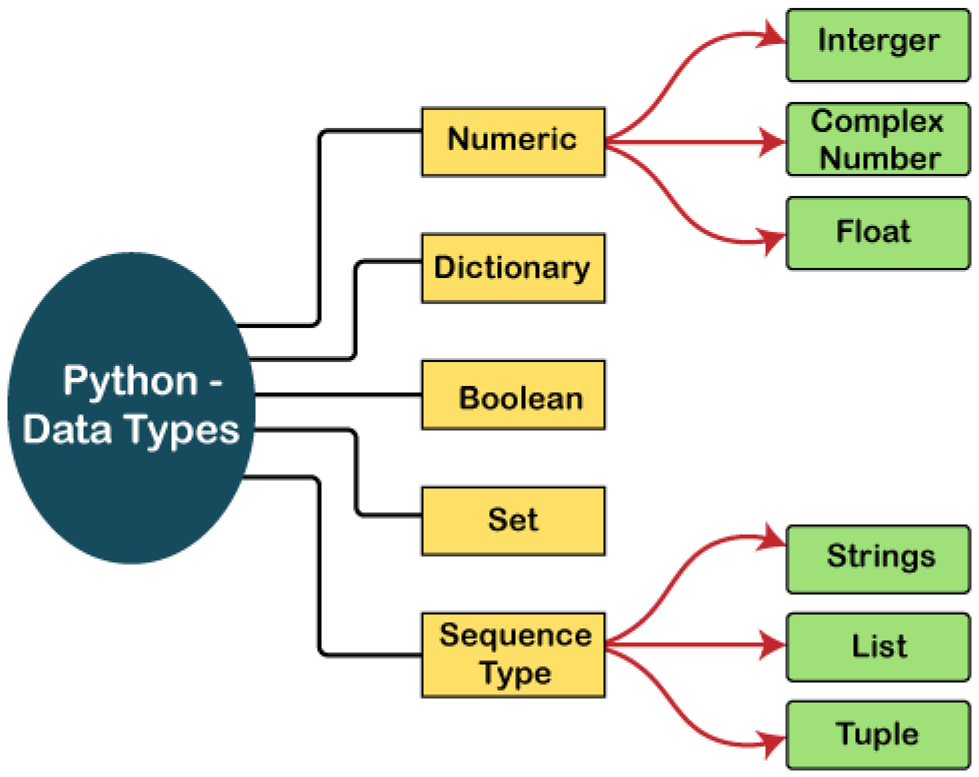
```
a=10
b="Hi Python"
c = 10.5
print(type(a))
print(type(b))
print(type(c))
```

Standard data types

1) A variable can hold different types of values. For example, a person's name must be stored as a string whereas its id must be stored as an integer.

2) Python provides various standard data types that define the storage method on each of them. The data types defined in Python are given below.

1. Numbers
2. Sequence Type
3. Boolean
4. Set
5. Dictionary



In this section of the tutorial, we will give a brief introduction of the above data-types. We will discuss each one of them in detail later in this tutorial.

1) Numbers

Number stores numeric values. The integer, float, and complex values belong to a Python Numbers data-type. Python provides the **type()** function to know the data-type of the variable. Similarly, the **isinstance()** function is used to check an object belongs to a particular class.

Python creates Number objects when a number is assigned to a variable. For example;

```
a = 5
print("The type of a", type(a))
b = 40.5
print("The type of b", type(b))
 c = 1+3j
print("The type of c", type(c))
print(" c is a complex number", isinstance(1+3j,complex))
```

Python supports three types of numeric data.

1. **Int -** Integer value can be any length such as integers 10, 2, 29, -20, -150 etc. Python has no restriction on the length of an integer. Its value belongs to **int**
2. **Float -** Float is used to store floating-point numbers like 1.9, 9.902, 15.2, etc. It is accurate upto 15 decimal points.
3. **complex** – complex numbers have a real and imaginary part, which are each implemented using double in C.

   A complex number contains an ordered pair, i.e., x + iy where x and y denote the real and imaginary parts, respectively. The complex numbers like 2.14j, 2.0 + 2.3j, etc.

2) Sequence Type

A) String

1) The string can be defined as the sequence of characters represented in the quotation marks. In Python, we can use single, double, or triple quotes to define a string.

2) String handling in Python is a straightforward task since Python provides built-in functions and operators to perform operations in the string.

3) In the case of string handling, the operator + is used to concatenate two strings as the operation *"hello"+" python"* returns *"hello python"*.

4) The operator * is known as a repetition operator as the operation "Python" *2 returns 'Python Python'.

The following example illustrates the string in Python.

Example - 1

```
str = "Hello Python"
print(str)
s = """A multiline
string"'
print(s)
```

B) List:

i) Python knows a number of compound data types, used to group together other values.
ii) Python Lists are similar to arrays in C. However, the list can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets [].
iii) We can use slice [:] operators to access the data of the list. The concatenation operator (+) and repetition operator (*) works with the list in the same way as they were working with the strings.

Consider the following example.

```
list1  = [1, "hi", "Python", 2]
print(type(list1))              #Checking type of given list

print (list1)                   #Printing the list1

print (list1[3:])               # List slicing

print (list1[0:2])              # List slicing

print (list1 + list1)           # List Concatenation using + operator

print (list1 * 3)               # List repetation using * operator
```

C) Tuple:

A tuple is similar to the list in many ways. Like lists, tuples also contain the collection of the items of different data types.

The items of the tuple are separated with a comma (,) and enclosed in parentheses ().

A tuple is a read-only data structure as we can't modify the size and value of the items of a tuple.

Let's see a simple example of the tuple.

```
tup  = ("hi", "Python", 2)
print (type(tup))                    # Checking type of tup
print (tup)                         #Printing the tuple

print (tup[1:])                     # Tuple slicing

print (tup[0:1])

print (tup + tup)                   # Tuple concatenation using + operator

print (tup * 3)                     # Tuple repatation using * operator

t[2] = "hi"
```

3) Dictionary:

Dictionary is an unordered set of a key-value pair of items.
It is like an associative array or a hash table where each key stores a specific value.
Key can hold any primitive data type, whereas value is an arbitrary Python object.
The items in the dictionary are separated with the comma (,) & enclosed in the curly braces {}.

Consider the following example.

```
d = {1:'Neer', 2:'Pratyush', 3:'Devansh', 4:'Reyansh'}

print (d)                                   # Printing dictionary

print("1st name is "+d[1])                   # Accesing value using keys
print("2nd name is "+ d[4])
print (d.keys())
print (d.values())
```

4) Boolean:

Boolean type provides two built-in values, True and False.
These values are used to determine the given statement true or false. It denotes by the class bool.
True can be represented by any non-zero value or 'T' whereas false can be represented by the 0 or 'F'.

Consider the following example.

```
# Python program to check the boolean type
print(type(True))
print(type(False))
print(false)
```

5) Set

Python Set is the unordered collection of the data type. It is iterable, mutable(can modify after creation), and has unique elements. In set, the order of the elements is undefined; it may return the changed sequence of the element. The set is created by using a built-in function **set(),** or a sequence of elements is passed in the curly braces and separated by the comma. It can contain various types of values. Consider the following example.

```
set1 = set()                        # Creating Empty set
set2 = {'James', 2, 3,'Python'}
print(set2)                              #Printing Set value
set2.add(10)                                # Adding element to the set
print(set2)                      #Removing element from the set
set2.remove(2)
print(set2)
```

Declaring Variable and Assigning Values

1) Python does not bind us to declare a variable before using it in the application. It allows us to create a variable at the required time.

2) We don't need to declare explicitly variable in Python. When we assign any value to the variable, that variable is declared automatically.

The equal (=) operator is used to assign value to a variable.

Object References

It is necessary to understand how the Python interpreter works when we declare a variable. The process of treating variables is somewhat different from many other programming languages.

Python is the highly object-oriented programming language; that's why every data item belongs to a specific type of class. Consider the following example.

```
print("Neer")
```

Output:

```
Neer
```

Object Identity

In Python, every created object identifies uniquely in Python. Python provides the guaranteed that no two objects will have the same identifier. The built-in **id()** function, is used to identify the object identifier. Consider the following example.

```
a = 50
b = a
print(id(a))
print(id(b))
# Reassigned variable a
a = 500
print(id(a))
```

Output:

```
140734982691168
140734982691168
2822056960944
```

We assigned the **b = a, a** and **b** both point to the same object. When we checked by the **id()** function it returned the same number. We reassign **a** to 500; then it referred to the new object identifier.

Variable Names

Variable names can be any length can have uppercase, lowercase (A to Z, a to z), the digit (0-9), and underscore character(_). Consider the following example of valid variables names.

```
name = "Pratyush"
age = 07
marks = 75.50

print(name)
print(age)
print(marks)
```

Output:

```
Pratyush
07
75.5
```

The multi-word keywords can be created by the following method.

- **Camel Case -** In the camel case, each word or abbreviation in the middle of begins with a capital letter. There is no intervention of whitespace. For example - nameOfStudent, valueOfVaraible, etc.
- **Pascal Case -** It is the same as the Camel Case, but here the first word is also capital. For example - NameOfStudent, etc.
- **Snake Case -** In the snake case, Words are separated by the underscore. For example - name_of_student, etc.

Multiple Assignment

Python allows us to assign a value to multiple variables in a single statement, which is also known as multiple assignments.

We can apply multiple assignments in two ways, either by assigning a single value to multiple variables or assigning multiple values to multiple variables. Consider the following example.

1. Assigning single value to multiple variables

Eg:

```
x=y=z=20
print(x)
print(y)
print(z)
```

Output:

```
20
20
20
```

2. Assigning multiple values to multiple variables:

Eg:

```
a,b,c=5,10,15
print a
print b
print c
```

Output:

```
5
10
15
```

The values will be assigned in the order in which variables appear.

Python Variable Types

There are two types of variables in Python - Local variable and Global variable. Let's understand the following variables.

Local Variable

Local variables are the variables that declared inside the function and have scope within the function. Let's understand the following example.

Example -

```
def add():                              # Declaring a function
    a = 20              # Defining local variables. They has scope only within a function
    b = 30
    c = a + b
     print("The sum is:", c)

   add()                             # Calling a function
```

Output:

```
        The sum is 50
```

Explanation:

In the above code, we declared a function named **add()** and assigned a few variables within the function. These variables will be referred to as the **local variables** which have scope only inside the function. If we try to use them outside the function, we get a following error.

Global Variables

Global variables can be used throughout the program, and its scope is in the entire program. We can use global variables inside or outside the function.

A variable declared outside the function is the global variable by default. Python provides the **global** keyword to use global variable inside the function. If we don't use the **global** keyword, the function treats it as a local variable. Let's understand the following example.

Example –

```
 x = 101                              # Declare a variable and initialize it
def mainFunction():                 # Global variable in function
  global x                            # printing a global variable
  print(x)

  x = 'Welcome To Python'             # modifying a global variable
  print(x)
 mainFunction()
  print(x)
```

Output:

```
101
Welcome  To Python
```

Explanation:

In the above code, we declare a global variable **x** and assign a value to it. Next, we defined a function and accessed the declared variable using the **global** keyword inside the function. Now we can modify its value. Then, we assigned a new string value to the variable x.

Now, we called the function and proceeded to print **x**. It printed the as newly assigned value of x.

Delete a variable

We can delete the variable using the **del** keyword. The syntax is given below.

Syntax –

del <variable_name>

In the following example, we create a variable x and assign value to it. We deleted variable x, and print it, we get the error **"variable x is not defined"**. The variable x will no longer use in future.

Example -

```
x = 6                          # Assigning a value to x
print(x)
del x                  # deleting a variable.
 print(x)
```

Maximum Possible Value of an Integer in Python

Unlike the other programming languages, Python doesn't have long int or float data types. It treats all integer values as an **int** data type. Here, the question arises. What is the maximum possible value can hold by the variable in Python? Consider the following example.

Example -

```
# A Python program to display that we can store  # large numbers in Python

a = 10000000000000000000000000000000000000000

a = a + 1

print(type(a))

print (a)
```

As we can see in the above example, we assigned a large integer value to variable **x** and checked its type. It printed **class <int>** not long int. Hence, there is no limitation number by bits and we can expand to the limit of our memory. Python doesn't have any special data type to store larger numbers.

Print Single and Multiple Variables in Python

We can print multiple variables within the single print statement. Below are the example of single and multiple printing values.

Example - 1 (Printing Single Variable)

```
a = 5                          # printing single value
print(a)
print((a))
```

Output:

Example - 2 (Printing Multiple Variables)

```
    a = 5
    b = 6                          # printing multiple variables
    print(a,b)
    Print(1, 2, 3, 4, 5, 6, 7, 8)          # separate the variables by the comma
```

Output:

```
            5 6
            1 2 3 4 5 6 7 8
```

Basic Fundamentals:

This section contains the fundamentals of Python, such as:

i)Tokens and their types.

ii) Comments

a)Tokens:

- The tokens can be defined as a punctuator mark, reserved words, and each word in a statement.
- The token is the smallest unit inside the given program.

There are following tokens in Python:

- Keywords.
- Identifiers.
- Literals.
- Operators.
- Consider the following explanation of keywords.

1. **True -** It represents the Boolean true, if the given condition is true, then it returns "True". Non-zero values are treated as true.
2. **False -** It represents the Boolean false; if the given condition is false, then it returns "False". Zero value is treated as false
3. **None -** It denotes the null value or void. An empty list or Zero can't be treated as **None**.
4. **and -** It is a logical operator. It is used to check the multiple conditions. It returns true if both conditions are true. Consider the following truth table.

5. or - It is a logical operator in Python. It returns true if one of the conditions is true. Consider the following truth table.

6. not - It is a logical operator and inverts the truth value. Consider the following truth table.

7. assert - This keyword is used as the debugging tool in Python. It checks the correctness of the code. It raises an **AssertionError** if found any error in the code and also prints the message with an error.

Example:

```
a = 10
b = 0
print('a is dividing by Zero')
assert b != 0
print(a / b)
```

8. def - This keyword is used to declare the function in Python. If followed by the function name.

```
def my_func(a,b):
   c = a+b
   print(c)
my_func(10,20)
```

9. class - It is used to represents the class in Python. The class is the blueprint of the objects. It is the collection of the variable and methods. Consider the following class.

```
class Myclass:
  #Variables……..
  def function_name(self):
    #statements………
```

10. continue - It is used to stop the execution of the current iteration. Consider the following example.

```
a = 0
while a < 4:
 a += 1
 if a == 2:
  continue
 print(a)
```

11. break - It is used to terminate the loop execution and control transfer to the end of the loop. Consider the following example.

Example

```
for i in range(5):
  if(i==3):
     break
  print(i)
print("End of execution")
```

12. If - It is used to represent the conditional statement. The execution of a particular block is decided by if statement. Consider the following example.

Example

```
i = 18
if (1 < 12):
print("I am less than 18")
```

13. else - The else statement is used with the if statement. When if statement returns false, then else block is executed. Consider the following example.

Example:

```
n = 11
if(n%2 == 0):
   print("Even")
else:
   print("odd")
```

14. elif - This Keyword is used to check the multiple conditions. It is short for **else-if**. If the previous condition is false, then check until the true condition is found. Condition the following example.

Example:

```
marks = int(input("Enter the marks:"))
if(marks>=90):
   print("Excellent")
elif(marks<90 and marks>=75):
   print("Very Good")
elif(marks<75 and marks>=60):
   print("Good")
else:
   print("Average")
```

15. del - It is used to delete the reference of the object. Consider the following example.

Example:

```
a=10
b=12
del a
print(b)
# a is no longer exist
print(a)
```

16. try, except - The try-except is used to handle the exceptions. The exceptions are run-time errors. Consider the following example.

Example:

```
a = 0
try:
  b = 1/a
except Exception as e:
  print(e)
```

17. raise - The raise keyword is used to through the exception forcefully. Consider the following example.

Example

```
a = 5
if (a>2):
  raise Exception('a should not exceed 2 ')
```

18. finally - The **finally** keyword is used to create a block of code that will always be executed no matter the else block raises an error or not. Consider the following example.

Example:

```
a=0
b=5
try:
   c = b/a
   print(c)
except Exception as e:
   print(e)
finally:
   print('Finally always executed')
```

19. for, while - Both keywords are used for iteration. The **for** keyword is used to iterate over the sequences (list, tuple, dictionary, string). A while loop is executed until the condition returns false. Consider the following example.

Example: For loop

```
list = [1,2,3,4,5]
for i in list:
   print(i)
```

Example: While loop

```
a = 0
while(a<5):
   print(a)
   a = a+1
```

20. import - The import keyword is used to import modules in the current Python script. The module contains a runnable Python code.

Example:

```
import math
print(math.sqrt(25))
```

21. from - This keyword is used to import the specific function or attributes in the current Python script.

Example:

```
from math import sqrt
print(sqrt(25))
```

22. as - It is used to create a name alias. It provides the user-define name while importing a module.

Example:

```
import calendar as cal
print(cal.month_name[5])
```

23. pass - The **pass** keyword is used to execute nothing or create a placeholder for future code. If we declare an empty class or function, it will through an error, so we use the pass keyword to declare an empty class or function.

Example:

```
   class my_class:
      pass

def my_func():
   pass
```

24. return - The **return** keyword is used to return the result value or none to called function.

Example:

```
def sum(a,b):
   c = a+b
   return c

print("The sum is:",sum(25,15))
```

25. is - This keyword is used to check if the two-variable refers to the same object. It returns the true if they refer to the same object otherwise false. Consider the following example.

Example

```
x = 5
y = 5

a = []
b = []
print(x is y)
print(a is b)
```

26. global - The global keyword is used to create a global variable inside the function. Any function can access the global. Consider the following example.

Example

```
def my_func():
   global a
   a = 10
   b = 20
   c = a+b
   print(c)

my_func()

def func():
   print(a)

func()
```

27. nonlocal - The **nonlocal** is similar to the **global** and used to work with a variable inside the nested function(function inside a function). Consider the following example.

Example

```
def outside_function():
   a = 20
   def inside_function():
      nonlocal a
      a = 30
      print("Inner function: ",a)
```

```
    inside_function()
    print("Outer function: ",a)
outside_function()
```

28. lambda - The lambda keyword is used to create the anonymous function in Python. It is an inline function without a name. Consider the following example.

Example

```
a = lambda x: x**2
for i in range(1,6):
  print(a(i))
```

29. yield - The **yield** keyword is used with the Python generator. It stops the function's execution and returns value to the caller. Consider the following example.

Example

```
def fun_Generator():
  yield 1
  yield 2
  yield 3

  # Driver code to check above generator function
for value in fun_Generator():
  print(value)
```

30. with - The **with** keyword is used in the exception handling. It makes code cleaner and more readable. The advantage of using **with**, we don't need to call **close()**. Consider the following example.

Example

```
with open('file_path', 'w') as file:
    file.write('hello world !')
```

31. None - The None keyword is used to define the null value. It is remembered that **None** does not indicate 0, false, or any empty data-types. It is an object of its data type, which is Consider the following example.

Example:

```
def return_none():
  a = 10
  b = 20
  c = a + b

x = return_none()
print(x)
```

We have covered all Python keywords. This is the brief introduction of Python Keywords. We will learn more in the upcoming tutorials.

Python IDEs

IDE stands for Integrated Development Environment is defined as a coding tool that helps to automate the process of editing, compiling, testing, etc. in an SDLC and it provides ease to the developer to run, write and debug the code.

It is specially designed for software development that consists of several tools which is used for developing and testing the software.

There are some Python IDEs which are as follows:

- PyCharm
- Spyder
- PyDev
- Atom
- Wing
- Jupyter Notebook

- Thonny
- Rodeo
- Microsoft Visual Studio
- Eric

PyCharm

PyCharm was developed by the Jet Brains, and it is a cross-platform Integrated Development Environment (IDE) specially designed for python. It is the most widely used IDE and available in both paid version and free open-source as well. It saves ample time by taking care of routine tasks.

Nested Structure in C

Keep Watching

It is a complete python IDE that is loaded with a rich set of features like auto code completion, quick project navigation, fast error checking and correction, remote development support, database accessibility, etc.

Features

- Smart code navigation
- Errors Highlighting
- Powerful debugger
- Supports Python web development frameworks, i.e., Angular JS, Javascript

---

Spyder

Spyder

Spyder is an open-source that has high recognition in the IDE market and most suitable for data science. The full name of Spyder is Scientific Python Development Environment. It supports all the significant platforms Linux, Windows, and MacOS X.

It provides a set of features like localized code editor, document viewer, variable explorer, integrated console, etc. and supports no. of scientific modules like NumPy, SciPy, etc.

Features

- Proper syntax highlighting and auto code completion
- Integrates strongly with IPython console
- Performs well in multi-language editor and auto code completion mode

---

PyDev

PyDev

PyDev is defined as one of the commonly used Python IDE, which is an external plugin for Eclipse. It is a natural choice of the Python developers that are coming from the Java background and very popular in the market as Python interpreter.

Aleksandar Totic is famous for his contribution to Mosaic browser and worked on Pydev project during 2003-2004.

Pydev has a feature which includes Django integration, automatic code completion, smart indents and block indents, etc.

Features

- o Strong Parameters like refactoring, debugging, code analysis, and code coverage function.
- o It supports virtual environments, Mypy, and black formatter.
- o Also supports PyLint integration, remote debugger, Unit test integration, etc.

Atom

Atom is developed by GitHub, which is initially started as an open-source, cross-platform. It is based on a framework, i.e., Electron which enables cross-platform desktop application using Chromium and Node.js and generally known as "Hackable Text Editor for the 21st century".

Features

- o Visualize the results on Atom without open any other window.
- o A plugin named "Markdown Preview Plus" provides built-in support for editing and visualizing Markdown files.

Wing

It is defined as a cross-platform IDE that is packed with necessary features and with decent development support. Its personal edition is free of cost. The pro version comes with a 30 days trial for the developers to try it out.

It has several features that include auto-completion, syntax highlighting, indents, and debugging.

Features

- Customizable and can have extensions as well.
- Supports remote development, test-driven development along with the unit test.

Jupyter Notebook

Jupyter is one of the most used IPython notebook editors that is used across the Data Science industry. It is a web application that is based on the server-client structure and allows you to create and manipulate notebook documents. It makes the best use of the fact that python is an interpreted language.

Features

- Supports markdowns
- Easy creation and editing of codes
- Ideal for beginners in data science

Thonny

Thonny is another IDE which is best suited for learning and teaching programming. It is a software developed at the University of Tartu and supports code completion and highlight syntax errors.

Features

- Simple debugger
- Supports highlighting errors and auto code completion

Rodeo

Rodeo is defined as one of the best IDE for python that is most widely used for data science projects like taking data and information from different resources.

It supports cross-platform functionality and provides auto-completion of code.

Features

- Allows the functions for comparing data, interact, plot, and inspect data.
- Auto code completion, syntax highlighter, visual file navigator, etc.

Microsoft Visual Studio

Microsoft Visual Studio is an open-source code editor which was best suited for development and debugging of latest web and cloud projects. It has its own marketplace for extensions.

Features

- Supports Python Coding in Visual studio

- Available in both paid and free version

Eric Python

The Eric Python is an editor which is developed in Python itself and can be used for both professional and non-professional work.

Features

- Offers configurable window layout, editors, source code folding
- Advanced project management capability, version control
- In-built debugger and task management support