Experiment 1

AIM : Introduction to Python

To understand basic syntax of Python programming language

THEORY :

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

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

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

SYNTAX :

- **Printing a String**

```
In [2]: print("Hello World!")
```

```
Hello World!
```

- **Lines and Indentation** Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

```
In [3]: if 6 > 4:
            print("Block Must be idented like this")
```

```
Block Must be idented like this
```

- **Multi-Line Statements** Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. Statements contained within the [], {}, or () brackets do not need to use the line continuation character.

In [4]:
```
string = 'Hello ' + \
         'World ' + \
         '!'
```

- **Comments in Python** A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

In [5]:
```
# print("This is a Comment")
print("This not a Comment")
```

```
This not a Comment
```

- **Variables and Value** Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

In [6]:
```
x = 5
y = 'Prince Piyush'
print(x, y)
```

```
5 Prince Piyush
```

- **Lists** - Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

In [7]:
```
x = ['Date', 12, 'height', 180, 5.10]
```

- **Tuples** - A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

In [8]:
```
x = ('banana', 'orange', 'grapes')
```

- **Dictionary** Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

In [9]:
```
Dict = {'one' : 1,
        'two' : 2,
        'three' : 3}
```

- **Loops** A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string). This is less like the for keyword in other programming language, and works more like an iterator method as found in other object-orientated programming languages.

In [10]:
```
fruits = ['banana', 'orange', 'grapes']
for fruit in fruits:
    print(fruit)
```

```
banana
orange
grapes
```

- **Functions** - A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.

In [11]:
```
def function(arguments):
    string = " Statements"
    print(string)
```

- **Lambda Fumctions** - A lambda function is a small anonymous function. A lambda function can take any number of arguments, but can only have one expression.

In [12]:
```
x = lambda a : a+10
print(x(10))
```

```
20
```

LEARNING OUTCOMES :

In this Experiment, We learned about basic syntax of Python programming language.