Programming with Python

What is Python?

- **Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.**
- It is used for:

1. web development (server-side),
2. software development,
3. mathematics,
4. system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

Python Syntax compared to other programming languages

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

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.

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.

Cont…

3) Interpreted Language

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) Cross-platform Language

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 is freely available on its official website **www.python.org**

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

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.

11) Embeddable

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.

12) Dynamic Memory Allocation

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.

Usage of Python

- Desktop Applications
- Web Applications
- Data Science
- Artificial Intelligence
- Machine Learning
- Scientific Computing
- Robotics
- Internet of Things (IoT)
- Gaming
- Mobile Apps
- Data Analysis and Preprocessing

Python Implementation

https://www.python.org/download/alternatives/

This site hosts the "traditional" implementation of Python (nicknamed CPython). A number of alternative implementations are available as well, namely

IronPython (Python running on .NET)

Jython (Python running on the Java Virtual Machine)

PyPy (A fast python implementation with a JIT compiler)

Stackless Python (Branch of CPython supporting microthreads)

MicroPython (Python running on micro controllers)

Python Version List

Python Version	Released Date
Python 1.0	January 1994
Python 1.5	December 31, 1997
Python 1.6	September 5, 2000
Python 2.0	October 16, 2000
Python 2.1	April 17, 2001
Python 2.2	December 21, 2001
Python 2.3	July 29, 2003
Python 2.4	November 30, 2004
Python 2.5	September 19, 2006
Python 2.6	October 1, 2008
Python 2.7	July 3, 2010
Python 3.0	December 3, 2008
Python 3.1	June 27, 2009
Python 3.2	February 20, 2011
Python 3.3	September 29, 2012
Python 3.4	March 16, 2014
Python 3.5	September 13, 2015
Python 3.6	December 23, 2016
Python 3.7	June 27, 2018
Python 3.8	October 14, 2019

Python Development Environments

What is Your Level?

- **Beginner** — IDLE (or Online Python Editors) is perfect choice for the first steps in python language. PyCharm is also good but takes the help of some experienced person while using this.
- **Intermediate** — PyCharm, Sublime, Atom, Vs Code.
- **Advanced** — PyCharm, Vim, Emacs, Sublime, Atom, Vs Code.

What's Your End Goal?

- **Web development** — PyCharm Professional, VS Code
- **Data Science** — Spyder, Jupyter Notebook, PyCharm Professional
- **Scripting** — Sublime, Atom, PyCharm Community, Eclipse + PyDev
- **QA** — Sublime, Atom, PyCharm Community, Jupyter Notebook

Python Indentation Rules

- Python uses 4 spaces as default indentation spaces. However, the number of spaces can be anything, it is up to the user. But a minimum of one space is needed to indent a statement.
- The first line of python code cannot have Indentation.
- Indentation is mandatory in python to define the blocks of statements.
- The number of spaces must be uniform in a block of code.
- It is preferred to use whitespaces instead of tabs to indent in python. Also, either use whitespace or tabs to indent, intermixing of tabs and whitespaces in indentation can cause wrong indentation errors.

Example:

```
name = 'Rahul'

if name == 'Rahul':
    print('WelCome Rahul..')
    print('How are you?')
else:
    print('Dude! whoever you are ')
    print('Why you here?')

print('Have a great day!')
```

Output:

```
WelCome Rahul..
How are you?
Happy Coding!
```

```
i = 1
while(i <= 6):
    print("Value is " + str(i))
    i = i + 1
```

Output:

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
Value is 6
```

```
number = 50

if(number != 0):
 if(number % 2 == 0):
   print("Given number is Even")
 else:
   print("Given number is Odd")
else:
 print("Given number is neither even nor odd")
```

Output:

```
Given number is Even
```

Wrong Indentation(Error):

```
if( 1 == 1):
print("This is test code")
```

With Correct Indentation:

```
if( 1 == 1):
  print("This is test code")
```

Wrong Indentation(Error):

```
if( 1 == 1):
 print("This is test code")
     print("This is test code1")
```

With Correct Indentation:

```
if( 1 == 1):
 print("This is test code")
 print("This is test code1")
```

Wrong Indentation(Error):

```
if( 1 == 1):
print("This is test code")
print("This is test code1")
```

With Correct Indentation:

```
if( 1 == 1):
 print("This is test code")
 print("This is test code1")
```

Wrong Indentation(Error):

```
if( 1 == 1):
   print("This is test code")

 print("This is test code1")
```

Correct Indentation:

```
if( 1 == 1):
   print("This is test code")

   print("This is test code1")
```

Benefits of Indentation in Python

- Indentation of code leads to better readability, although the primary reason for indentation in python is to identify block structures.
- Missing { and } errors that sometimes popup in c,c++ languages can be avoided in python, also the number of lines of code is reduced.

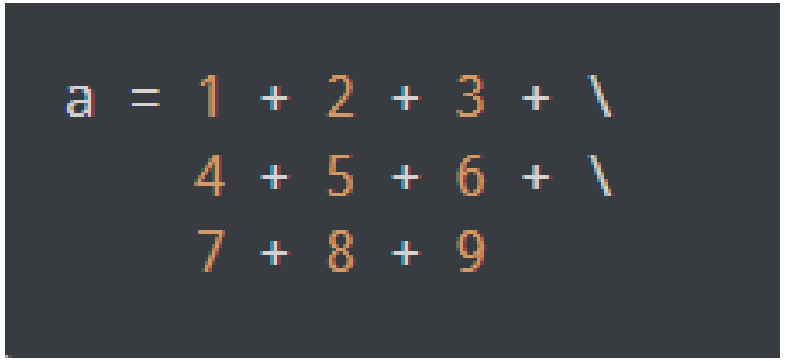
Disadvantages of indentation in Python

- Code must be carefully indented with proper no of whitespace and making sure that uniformity of whitespaces is maintained in a single block. If the number of lines in python code is huge, sometimes this can become tedious if by chance the indentation is corrupted.
- Without the use of good editors/Ide's that help with the indentation, writing a python code, especially huge lines of code is sometimes a tedious task, because, for each line, we should make a type in the indentation as well.

Python Statement

- **Multi-line statement**

In Python, the end of a statement is marked by a newline character. But we can make a statement extend over multiple lines with the line continuation character (\). For example:



- This is an explicit line continuation. In Python, line continuation is implied inside parentheses ( ), brackets [ ], and braces { }. For instance, we can implement the above multi-line statement as:

```
a = (1 + 2 + 3 +
     4 + 5 + 6 +
     7 + 8 + 9)
```

```
colors = ['red',
          'blue',
          'green']
```

- We can also put multiple statements in a single line using semicolons, as follows:

```
a = 1; b = 2; c = 3
```

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.
- The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:
   print "True"
else:
   print "False"
```

Valid code

```
if True:
print "Answer"
print "True"
else:
print "Answer"
print "False"
```

Invalid code

Comments

- Comments starts with a #, and Python will ignore them:

```
#This is a comment.

print("Hello, World!")
```

Comments can be placed at the end of a line, and Python will ignore the rest of the line:

```
print("Hello, World!") #This is a comment
```

- **Multi-line comments**

We can have comments that extend up to multiple lines. One way is to use the hash(**#**) symbol at the beginning of each line. For example:

```
#This is a comment
#written in
#more than just one line
print("Hello, World!")
```

Another way of doing this is to use triple quotes, either ’ ’ ’ or ” ” ”.

These triple quotes are generally used for multi-line strings. But they can be used as a multi-line comment as well. Unless they are not docstrings, they do not generate any extra code.

```
"""
This is a comment
written in
more than just one line
"""

print("Hello, World!")
```

Character set

- Python character set is a valid set of characters recognized by the Python language.
- **Alphabets:** All capital (A-Z) and small (a-z) alphabets.
- **Digits:** All digits 0-9.
- **Special Symbols:** Python supports all kind of special symbols like, ” ‘ l ; : ! ~ @ # $ % ^ ` & * ( ) _ + – = { } [ ] \ .
- **White Spaces:** White spaces like tab space, blank space, newline, and carriage return.
- **Other:** All ASCII and UNICODE characters are supported by Python that constitutes the Python character set.

Tokens

- A token is the smallest individual unit in a python program. All statements and instructions in a program are built with tokens. The various tokens in python are :

1. Keywords
2. Identifiers
3. Literals or Values
4. Operators
5. Punctuators

Keywords

- Keywords are the reserved words in Python.
- We cannot use a keyword as a variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language.
- In Python, keywords are case sensitive.
- There are 33 keywords in Python 3.7. This number can vary slightly over the course of time.

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

Identifiers

- An identifier is a name given to entities like class, functions, variables, etc. It helps to differentiate one entity from another.
- Rules for writing identifier:
- Python is case-sensitive. So case matters in naming identifiers. And hence **python** and **Python** are two different identifiers.
- Identifier starts with a capital letter (A-Z) , a small letter (a-z) or an underscore(_). It can't start with any other character.
- Except for letters and underscore, digits can also be a part of identifier but can't be the first character of it.
- Any other special characters or whitespaces are strictly prohibited in an identifier.
- An identifier can't be a keyword.

1.

Literals or values

- Literals are the fixed values or data items used in a source code. Python supports different types of literals such as:
- **(i) String Literals:** The text written in single, double, or triple quotes represents the string literals in Python. For example: “Computer Science”, ‘sam’, etc. We can also use triple quotes to write multi-line strings.

Example:

```
# String Literals
a = 'Hello'
b = “CGPIT"
c = ”CGPIT is a
        learning platform'"


# Driver code
print(a)
print(b)
print(c)
```

- **(ii) Character Literals:** Character literal is also a string literal type in which the character is enclosed in single or double-quotes.

```
# Character Literals
a = 'G'
b = "W"

# Driver code
print(a)
print(b)
```

- **(iii) Numeric Literals:** These are the literals written in form of numbers. Python supports the following numerical literals:
- **Integer Literal:** It includes both positive and negative numbers along with 0. It doesn't include fractional parts. It can also include binary, decimal, octal, hexadecimal literal.
- **Float Literal:** It includes both positive and negative real numbers. It also includes fractional parts.
- **Complex Literal:** It includes a+bi numeral, here a represents the real part and b represents the complex part.

```
# Numeric Literals
a = 5
b = 10.3
c = -17

# Driver code
print(a)
print(b)
print(c)
```

- **(iv) Boolean Literals:** Boolean literals have only two values in Python. These are True and False.

```
# Boolean Literals

a = 3

b = (a == 3)

c = True + 10


# Driver code

print(a, b, c)
```

Output:

3 True 11

- **(v) Special Literals:** Python has a special literal ‘None’. It is used to denote nothing, no values, or the absence of value.

- # Special Literals
- var = None
- print(var)

- **(vi) Literals Collections:** Literals collections in python includes list, tuple, dictionary, and sets.
- **List:** It is a list of elements represented in square brackets with commas in between. These variables can be of any data type and can be changed as well.
- **Tuple:** It is also a list of comma-separated elements or values in round brackets. The values can be of any data type but can't be changed.
- **Dictionary:** It is the unordered set of key-value pairs.
- **Set:** It is the unordered collection of elements in curly braces '{}'.

- **4. Operators:** These are the tokens responsible to perform an operation in an expression. The variables on which operation is applied are called *operands*. Operators can be unary or binary. Unary operators are the ones acting on a single operand like complement operator, etc. While binary operators need two operands to operate.

```
# Operators
a = 12

# Unary operator
b = ~ a

# Binary operator
c = a+b

# Driver code
print(b)
print(c)
```

- **5. Punctuators:** These are the symbols that used in Python to organize the structures, statements, and expressions. Some of the Punctuators are: [ ] { } ( ) @ -= += *= //= **== = , etc.