History of Python

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 was **conceived in the late 1980s by Guido van Rossum** at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989.

Features in Python

- Easy to code: Python is a high-level programming language. ...
- Free and Open Source: ...
- Object-Oriented Language: ...
- GUI Programming Support: ...
- High-Level Language: ...
- Extensible feature: ...
- Python is Portable language: ...
- Python is Integrated language:

Structuring Python Programs

Python statements are written in such a format that one statement is only written in a single line. The interpreter considers the 'new line character' as the terminator of one instruction. But, writing multiple statements per line is also possible that you can find below.
Examples:

```
# Example 1

print('Welcome to Geeks for Geeks')
```

Output:

```
Welcome to Geeks for Geeks
```

Multiple Statements per Line

We can also write multiple statements per line, but it is not a good practice as it reduces the readability of the code. Try to avoid writing multiple statements in a single line. But, still you can write multiple lines

by terminating one statement with the help of ';'. ';' is used as the terminator of one statement in this ase. For Example, consider the following code.

```
# Example

a = 10; b = 20; c = b + a

print(a); print(b); print(c)
```

Output:

```
10
20
30
```

Line Continuation to avoid left and right scrolling

Some statements may become very long and may force you to scroll the screen left and right frequently. You can fit your code in such a way that you do not have to scroll here and there. Python allows you to write a single statement in multiple lines, also known as line continuation. Line continuation enhances readability as well.

```
# Bad Practice as width of this code is too much.


#code

x = 10

y = 20

z = 30

no_of_teachers = x

no_of_male_students = y

no_of_female_students = z


if (no_of_teachers == 10 and no_of_female_students == 30 and
no_of_male_students == 20 and (x + y) == 30):

    print('The course is valid')


# This could be done instead:


if (no_of_teachers == 10 and no_of_female_students == 30
```

```
    and no_of_male_students == 20 and x + y == 30):
    print('The course is valid')
```

Types of Line Continuation

In general, there are two types of line continuation

- **Implicit Line Continuation**

  This is the most straightforward technique in writing a statement that spans multiple lines.

  Any statement containing opening parentheses (‘(‘), brackets (‘[‘), or curly braces (‘{‘) is presumed to be incomplete until all matching parentheses, square brackets, and curly braces have been encountered. Until then, the statement can be implicitly continued across lines without raising an error.

  Examples:

```
# Example 1

# The following code is valid
a = [
    [1, 2, 3],
    [3, 4, 5],
    [5, 6, 7]
    ]

print(a)
```

- **Output:**
- `[[1, 2, 3], [3, 4, 5], [5, 6, 7]]`

Explicit Line Continuation

Explicit Line joining is used mostly when implicit line joining is not applicable. In this method, you have to use a character that helps the interpreter to understand that the particular statement is spanning more than one lines.

Backslash (\) is used to indicate that a statement spans more than one line. The point is to be noted that ” must be the last character in that line, even white-space is not allowed.

See the following example for clarification

```
# Example

x = \
    1 + 2 \
    + 5 + 6 \
    + 10

print(x)
```

- **Output:**
- 24

- **Comments in Python**
  Writing comments in the code are very important and they help in code readability and also tell more about the code. It helps you to write details against a statement or a chunk of code. Interpreter ignores the comments and does not count them in commands. In this section, we'll learn how to write comments in Python.

  Symbols used for writing comments include Hash (#) or Triple Double Quotation marks(“””). Hash is used in writing single line comments that do not span multiple lines. Triple Quotation Marks are used to write multiple line comments. Three triple quotation marks to start the comment and again three quotation marks to end the comment.
  Consider the following examples:

```
# Example 1



####### This example will print Hello World ####### print('Hello
World')  # This is a comment



# Example 2

""" This example will demonstrate
    multiple comments """

""" The following
    a variable contains the
    string 'How old are you?'
"""
a = 'How old are you?'

""" The following statement prints
    what's inside the variable a
"""
print(a)
```

- **Note** Do note that Hash (#) inside a string does not make it a comment. Consider the following example for demonstration.

```
# Example

""" The following statement prints the string stored
    in the variable """

a = 'This is # not a comment #'
print(a) # Prints the string stored in a
```

- **White spaces**
  The most common whitespace characters are the following:

Character	ASCII Code	Literal Expression
Space	32 (0x20)	‘ ‘
tab	9 (0x9)	‘\t’
newline	10 (0xA)	‘\n’

- * You can always refer to ASCII Table by clicking here.

  Whitespace is mostly ignored, and mostly not required, by the Python interpreter. When it is clear where one token ends and the next one starts, whitespace can be omitted. This is usually the case when special non-alphanumeric characters are involved.
  Examples:

```
# Example 1

# This is correct but whitespace can improve readability

a = 1-2 # Better way is a = 1 - 2

print(a)
```

- Whitespaces are necessary in separating the keywords from the variables or other keywords. Consider the following example.

```
# Example

x = [1, 2, 3]
y = 2

""" Following is incorrect, and will generate syntax error
a = yin x
"""

# Corrected version is written as
a = y in x
print(a)
```

- **Whitespaces as Indentation**
  Python’s syntax is quite easy, but still you have to take some care in writing the code. Indentation is used in writing python codes.

Whitespaces before a statement have significant role and are used in indentation. Whitespace before a statement can have a different meaning. Let's try an example.

```
# Example

print('foo') # Correct

   print('foo') # This will generate an error

# The error would be somewhat 'unexpected indent'
```

- Leading whitespaces are used to determine the grouping of the statements like in loops or control structures etc.
  Example:

```
# Example

x = 10

while(x != 0):
 if(x > 5):   # Line 1
  print('x > 5')  # Line 2
 else:        # Line 3
  print('x < 5') # Line 4
 x -= 2    # Line 5

"""
Lines 1, 3, 5 are on same level
Line 2 will only be executed if if condition becomes true.
Line 4 will only be executed if if condition becomes false.
"""
```

- **Output:**
- x > 5
- x > 5
- x > 5
- x < 5
- x < 5