Al-Mustaqbal University
College of Sciences
Intelligent Medical System Department

جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم
قسم الانظمة الطبية الذكية

Lab: (1)

Introduction to Python Syntax

Subject: Object oriented programming I

Class: Second

Lecturer: Dr. Maytham N. Meqdad

Introduction to Python Syntax

Python Syntax

Python is known for its clean and readable syntax, which makes it an excellent language for beginners, but also powerful enough for advanced applications. In this article, we will learn about the basic elements of Python syntax.

Prerequisites: Before diving into Python syntax, ensure you have:

- Install Python (preferably Python 3.x).
- A text editor or IDE (Integrated Development Environment) like VSCode, PyCharm, or even IDLE (Python's built-in IDE).

What is Python Syntax?

- Python syntax refers to the set of rules that defines the combinations of symbols that are considered to be correctly structured programs in the Python language. These rules make sure that programs written in Python should be structured and formatted, ensuring that the Python interpreter can understand and execute them correctly.

- Python Hello World

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

Output: `Hello, World!`

Indentation in Python

Python Indentation refers to the use of whitespace (spaces or tabs) at the beginning of a line of code in Python. It is used to define the code blocks. Indentation is crucial in Python because, unlike many other programming languages that use braces "{}" to define blocks, Python uses indentation. It improves the readability of Python code, but on other hand it became difficult to rectify indentation errors. Even one extra or less space can leads to identation error.

```
if 10 > 5:
    print("This is true!")

if 10 > 5:
    print("This is true!")
```

Python Variables

Variables in Python are essentially named references pointing to objects in memory. Unlike some other languages, in Python, you don't need to declare a variable's type explicitly. Based on the value assigned, Python will dynamically determine the type. In the below example, we create a variable 'a' and initialize it with interger value so, type of 'a' is int then we store the string value in 'a' it become 'str' type. It is called dynamic typing which means variable's data type can change during runtime.

Indentation in Python

Python Indentation refers to the use of whitespace (spaces or tabs) at the beginning of a line of code in Python. It is used to define the code blocks. Indentation is crucial in Python because, unlike many other programming languages that use braces "{}" to define blocks, Python uses indentation. It improves the readability of Python code, but on other hand it became difficult to rectify indentation errors. Even one extra or less space can leads to identation error.

```
if 10 > 5:
    print("This is true!")

if 10 > 5:
    print("This is true!")
```

Python keywords

Keywords in Python are reserved words that have special meanings and serve specific purposes in the language syntax. They cannot be used as identifiers (names for variables, functions, classes, etc.). Below is the list of keywords in Python:

```
False       await         else          import          pass
None       break        except     in                   raise
True        class            finally      is                    return
and         continue    for           lambda         try
as            def             from        nonlocal       while
assert     del             global      not                with
async      elif             if              or                  yield
```

With different versions of Python some keywords may vary. We can see all the keywords of current version of Python using below code.

```
import keyword
 #printing all keywords at once using "kwlist"()
print("The list of keywords is(" :
print(keyword.kwlist)
```

Python Single Line Comment

Single line comments are preceded by the "#" symbol. Everything after this symbol on the same line is considered a comment and not considered as the part of execution of code. Below is the example of single line comment and we can see that there is no effect of comment on output.

```
first_name = "Reddy"
last_name = "Anna"

# print full name
print(first_name, last_name)
```

Output

```
Reddy Anna
```

Python Multi-line Comment

Python doesn't have a specific syntax for multi-line comments. However, programmers often use multiple single-line comments, one after the other, or sometimes triple quotes (either ''' or """), even though they're technically string literals. Below is the example of multiline comment.

```
# This is a Python program
# to explain multiline comment.

'''
Functions to print table of
any number.
'''
def print_table(n):
  for i in range(1,11):
    print(i*n)

print_table(4)
```

Output

```
4
8
12
16
20
24
28
32
36
40
```

Using Backslashes (\)

In Python, you can break a statement into multiple lines using the backslash (\). This method is useful, especially when we are working with strings or mathematical operations.

```
sentence = "This is a very long sentence that we want to " \
           "split over multiple lines for better readability."

print(sentence)

# For mathematical operations
total = 1 + 2 + 3 + \
        4 + 5 + 6 + \
        7 + 8 + 9

print(total)
```

Output

```
This is a very long sentence that we want to split over multiple lines for
better readability.
45
```

Using Parentheses

For certain constructs, like lists, tuples, or function arguments, we can split statements over multiple lines inside the parentheses, without needing backslashes.

```
# Create list
numbers = [1, 2, 3,
           4, 5, 6,
           7, 8, 9]

def total(num1, num2, num3, num4):
  print(num1+num2+num3+num4)
```

```
# Function call
total(23, 34,
      22, 21)
```

Output

```
100
```

Triple Quotes for Strings

If we are working with docstrings or multiline strings, we can use triple quotes (single ”’ or double “””).

```
text = """GeeksforGeeks Interactive Live and Self-Paced
Courses to help you enhance your programming.
Practice problems and learn with live and online
recorded classes with GFG courses. Offline Programs."""

print(text)
```

Output

```
GeeksforGeeks Interactive Live and Self-Paced
Courses to help you enhance your programming.
Practice problems and learn with live and online
recorded classes with GFG courses. Offline Programs.
```

Quotation in Python

In Python, strings can be enclosed using single (‘), double (“), or triple (”’ or “””) quotes. Single and double quotes are interchangeable for defining simple strings, while triple quotes allow for the creation of multiline strings. That we have used in above example. The choice of quotation type can simplify inserting one type of quote within a string without the need for escaping, for example, using double quotes to enclose a string that contains a single quote. Below is the example of using single and double quotes.

```
# Embedded single quote inside double.
text1 = "He said, 'I learned Python from GeeksforGeeks'"

# Embedded double quote inside single.
text2 = 'He said, "I have created a project using Python"'

print(text1)
print(text2)
```

Output

```
He said, 'I learned Python from GeeksforGeeks'
He said, "I have created a project using Python"
```

Continuation of Statements in Python

In Python, statements are typically written on a single line. However, there are scenarios where writing a statement on multiple lines can improve readability or is required due to the length of the statement. This continuation of statements over multiple lines is supported in Python in various ways:

Implicit Continuation

Python implicitly supports line continuation within parentheses (), square brackets [], and curly braces {}. This is often used in defining multi-line lists, tuples, dictionaries, or function arguments.

```
# Line continuation within square brackets '[]'
numbers = [
    1, 2, 3,
    4, 5, 6,
    7, 8, 9
]

# Line continuation within parentheses '()'
result = max(
    10, 20,
    30, 40
)

# Line continuation within curly braces '{}'
dictionary = {
    "name": "Alice",
    "age": 25,
    "address": "123 Wonderland"
}

print(numbers)
print(result)
print(dictionary)
```

Output

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
40
{'name': 'Alice', 'age': 25, 'address': '123 Wonderland'}
```

Explicit Continuation

If you're not inside any of the above constructs, you can use the backslash '\' to indicate that a statement should continue on the next line. This is known as explicit line continuation.

```
# Explicit continuation
s = "GFG is computer science portal " \
    "that is used by geeks."

print(s)
```

Output

```
GFG is computer science portal that that is used by geeks.
```

Note: Using a backslash does have some pitfalls, such as if there's a space after the backslash, it will result in a syntax error.

Strings

Strings can be continued over multiple lines using triple quotes (''' or """). Additionally, if two string literals are adjacent to each other, Python will automatically concatenate them.

```
text = '''A Geek can help other
Geek by writing article on GFG'''

message = "Hello, " "Geeks!"

print(text)
print(message)
```

Output

```
A Geek can help other
Geek by writing article on GFG
Hello, Geeks!
```

String Literals in Python

String literals in Python are sequences of characters used to represent textual data. Here's a detailed look at string literals in Python. String literals can be enclosed in single quotes ('), double quotes ("), or triple quotes (''' or """).

```
string1 = 'Hello, Geeks'
string2 = "Namaste, Geeks"

multi_line_string = '''Ram learned Python
by reading tutorial on
GeeksforGeeks'''

print(string1)
print(string2)
print(multi_line_string)
```

Output

```
Hello, Geeks
Namaste, Geeks
Ram learned Python
by reading tutorial on
GeeksforGeeks
```