Mizpah Christian School - https://www.mizpahchristianschool.org

---

Topic :Theory Python

Python Revision

Variable Naming Rules

- Variable names in Python may contain uppercase and lowercase letters (A–Z, a–z), digits (0–9), and underscores (_), but they cannot begin with a digit.
- Variables are created upon assignment.
- Variables can refer to objects of any type.

Examples of valid Python variable names:

- `string1`
- `_a1p4a`
- `list_of_names`

Examples of invalid variable names (start with a digit):

- `9lives`
- `99_balloons`
- `2beOrNot2Be`

> **Note:** Python variable names can contain Unicode characters, which allows for letters from non-English alphabets. However, to ensure compatibility across regions, avoid decorated characters when sharing code.

Keywords in Python

The following are reserved keywords in Python. They form the core commands of the language and cannot be used as variable names:

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

Jargon

- **Complex data types:** Structured types made from other data types.
- **Constant:** A value that does not change during program execution.
- **Constrain:** Ensuring calculation results stay within a specific range.
- **Hash:** A smaller, calculated number from a dictionary key for storage efficiency.
- **Immutable:** Values that cannot be changed in place.
- **Index:** A token used to point to a specific item in a sequence.
- **Iterable:** An object that can be iterated.
- **Iterate:** Looping through items in a sequence one at a time.
- **Iterator:** A construct designed for looping.
- **Mapping:** A sequence that maps hashable values to objects.
- **Matrix:** A multidimensional sequence.
- **Method:** A function attached to an object or class.
- **Mutable:** Values that can be changed.
- **Operation:** An action performed on two variables (operands), often mathematical or logical.
- **Queue:** A first-in, first-out (FIFO) structure.
- **Resultant:** The value returned as a result of a process.
- **Separator:** Text used to distinguish between items of data.
- **Sequence:** The simplest complex data type (e.g., lists, strings, tuples).
- **Sequence packing:** Assigning a sequence of values to a variable.
- **Slice:** A segment of a sequence.
- **Stack:** A last-in, first-out (LIFO) structure.
- **Traverse:** Going through the items in a sequence in order.

Overview of Python

Python is a high-level, general-purpose programming language. Its features make it suitable for various applications, including scripting, web development, and compute-intensive tasks.

- **Advantages of Python:**
  - Auto-compilation to bytecode.
  - Extendable in C and C++ for enhanced performance.
  - Strong structuring constructs and consistent object-oriented programming.

Key Features of Python

- **High-level data types:** Strings, lists, dictionaries, etc.
- **Control structures:** `if`, `if-else`, `while`, and powerful collection iterators like `for`.
- **Organizational structure:** Functions, classes, modules, and packages.
- **Bytecode compilation:** Source code is compiled to bytecode without a separate compilation step.
- **Object-oriented:** Everything in Python is an object, making it easy to implement classes.
- **Extensions in C/C++:** Tools like SWIG, sip, and Pyrex aid in writing extension modules.

Sequence Types in Python

- **Immutable types:**
  - **Tuples:** Can contain any data type.
  - **Strings:** Made of characters.
- **Mutable types:**
  - **Lists:** Can contain any data type.

Sample Code in Python - Predict the Output

```
x = 34 - 23  # A comment.
y = "Hello"  # Another one.
z = 3.45
if z == 3.45 or y == "Hello":
        x = x + 1
        y = y + " World"  # String concatenation.
print(x)
print(y)
```

Python Standard Data Types

Python has five standard data types:

- **Numbers**

- **String**
- **List**
- **Tuple**
- **Dictionary**

Predict the Data Types

- `var1 = 10`
- `var2 = 1`

Numerical Data Types

- **int**: Signed integers.
- **float**: Floating-point real values.
- **complex**: Complex numbers.

String Operations - Sample Code

```
str = 'Hello World!'
print(str)          # Prints complete string
print(str[0])       # Prints first character of the string
print(str[2:5])     # Prints characters from the 3rd to the 5th position
print(str[2:])      # Prints string starting from the 3rd character
print(str * 2)      # Prints the string two times
print(str + "TEST")  # Prints concatenated string
```

Output

```
Hello World!
H
llo
llo World!
Hello World!Hello World!
Hello World!TEST
```

List Operations - Sample Code

```
list = ['abcd', 786, 2.23, 'john', 70.2]
tinylist = [123, 'john']
```

```
print(list)             # Prints complete list
print(list[0])          # Prints first element of the list
print(list[1:3])        # Prints elements from the 2nd to 3rd positions
print(list[2:])         # Prints elements starting from the 3rd position
print(tinylist * 2)     # Prints the list two times
print(list + tinylist)  # Prints concatenated lists
```

Output:

```
['abcd', 786, 2.23, 'john', 70.2]
abcd
[786, 2.23]
[2.23, 'john', 70.2]
[123, 'john', 123, 'john']
['abcd', 786, 2.23, 'john', 70.2, 123, 'john']
```

Python Operators

Python supports the following types of operators:

- **Arithmetic Operators**: `+`, `-`, `*`, `/`, `%`, `**`, `//`
- **Comparison Operators**: `==`, `!=`, `>`, `<`, `>=`, `<=`
- **Assignment Operators**: `=`, `+=`, `-=`, `*=`, `/=`, `%=`, `**=`, `//=`
- **Logical Operators**: `and`, `or`, `not`
- **Bitwise Operators**: `&`, `|`, `^`, `~`, `<<`, `>>`
- **Membership Operators**: `in`, `not in`
- **Identity Operators**: `is`, `is not`

END