Day 2: Python Programming

- **Identifiers:**

In Python, identifiers are names used to identify variables, functions, classes, modules, and other objects.
Rules for naming identifiers in Python:

1. Identifiers must start with a letter (a-z, A-Z) or an underscore (_). They cannot start with a digit. Example: total123, _Hello234,etc. However, 123hello is not valid.
2. After the first character, identifiers can include letters (a-z, A-Z), digits (0-9), and underscores (_).
3. Identifiers are case sensitive.
   Example: myVariable, MyVariable, and MYVARIABLE are considered different identifiers.
4. There is no specific limit on the length of an identifier, but it is recommended to keep them short and descriptive.
5. Reserves/Keywords cannot be used as an identifier such as if, while, else, etc.

Note:
_a : denotes Private variable
__a : denotes Protected/ Strong private
__a__ : denotes Magic variable/ Dunder variable

- **Reserved Words/Keywords:**

Python has a set of reserved keywords that cannot be used as identifiers (names for variables, functions, classes, etc.). These keywords are reserved by the language syntax and have specific meanings in Python. The list of reserved keywords can be checked using the 'keyword' module:

```
import keyword
print(keyword.kwlist)
```

Output:

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await',
'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']
```

- **Data types:**

Data types give two information:

- Types of data, and
- Operations on those data

In Python, we do not need to explicitly define the data type as it is dynamically typed.
We have the following example:

```
a = 10
print(a)
print(type(a))
print(id(a))
a = 10.5
print(a)
print(type(a))
print(id(a))
```

Output:

```
10
<class 'int'>
4352044144
10.5
<class 'float'>
4343460496
```

Python contains 14 data types:

(5 fundamental data types)

1. int
2. float
3. complex
4. Bool
5. String
6. Byte
7. Bytearray
8. Range
9. List
10. Tuple
11. Set
12. Frozenset
13. Dict
14. None

1. int:

Incase of Python2, there were two types of int: int and long int. But with Python3, this concept has since been removed.
‘int’ is an integral number or whole number. Example:

```
a=10
print(type(a))
```

Output: `<class 'int'>`

Integer can be denoted in four forms in Python:

1. Binary form:
   Allowed characters: 0 and 1.
   We can show binary form as: 0b1010/0B1010.
   Example:
   ```
   a = 0b1010
   print(a)
   ```
   Output: `10` //since decimal form is default

2. Decimal form (default):
   Allowed characters: 0-9

3. Octal form:
   Allowed characters: 0-7
   Example:
   ```
   a = 0o2015
   print(a)
   ```
   Output: `1037` //since decimal form is default

4. Hexadecimal form:
   Allowed character: 0-9, A-F
   Example:
   ```
   a = 0x1032Beef
   print(a)
   ```
   Output: `271761135` //since decimal form is default

Base conversion methods:

- To binary: bin ()
  ```
  a = 0b1010
  print(bin(a))
  ```
  Output: `0b1010`

- To octal: oct()
  ```
  a = 0o2015
  print(oct(a))
  ```

Output: `0o2015`

- To hexadecimal: hex()

```
a = 0x1032Beef
print(hex(a))
```

Output: `0x1032beef`

Thus, we can see that the outputs are not in decimal in the above programs after base conversion.

2. **float:**

‘float’ is used to represent floating-point numbers, which are numbers with a decimal point.

Example 1:

```
a = 1.2
print(a)
```

Output: `1.2`

Example 2:

```
a=1.2e3
print(a)
```

Output: `1200.0`

Type conversion is not possible in float number, only decimal form is possible.
Let us consider the following example:

```
a=1.2
print(bin(a))
```

Output:

```
print(bin(a))
     ^^^^^^
TypeError: 'float' object cannot be interpreted as an integer
```