Day 14 Task: Python Data Types and Data Structures for DevOps

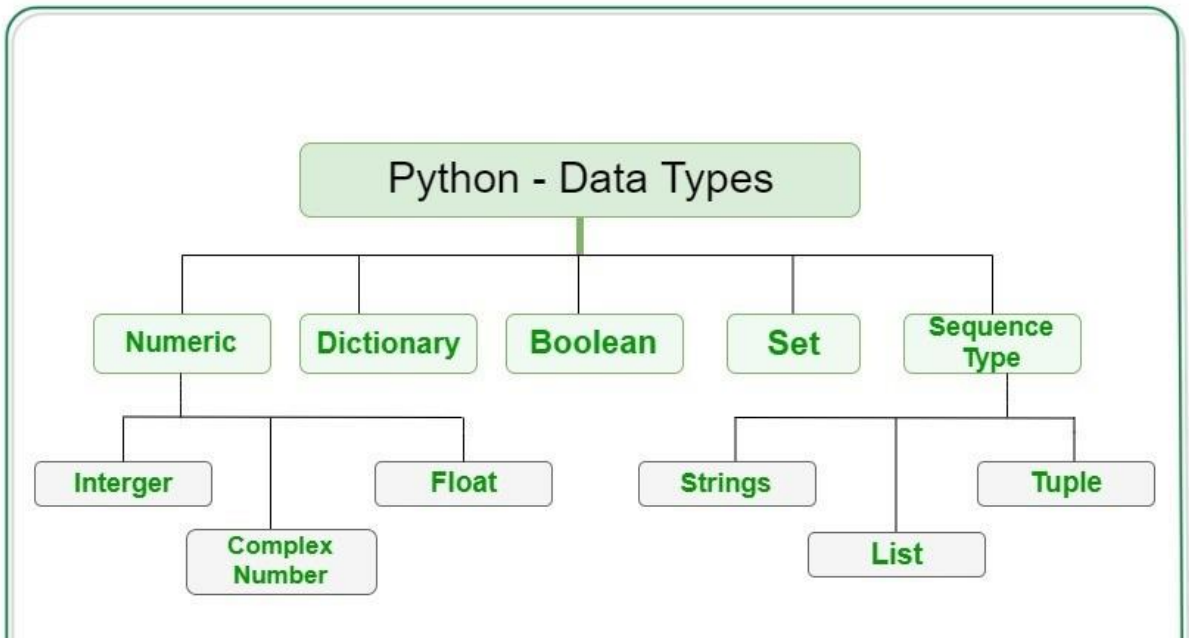
Data Types

Python Data Types are used to define the type of a variable. Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.

For example, a person's name is stored as a alpha characters and his age is stored as numeric value and address is stored as alphanumeric characters.

Following are the standard or built-in data type of Python:

- Numeric - int, float, complex
- String - str
- Sequence - list, tuple, range
- Binary - bytes, bytearray, memoryview
- Mapping - dict
- Boolean - bool
- Set - set, frozenset
- None - NoneType



Data Structures

Data Structures are a way of organizing data so that it can be accessed more efficiently depending upon the situation. Data Structures are fundamentals of any programming language around which a program is built. Python helps to learn the fundamental of these data structures in a simpler way as compared to other programming languages.

Data structures in Python are a way to organize and store data in a specific format, allowing for efficient manipulation and retrieval. Some common data structures in Python include:

- **Lists:** Lists are a collection of items in a specific order. They are defined using square brackets, and items can be accessed by their index number. Lists are mutable, meaning the contents can be changed.

- **Tuples:** Tuples are similar to lists, but they are immutable, meaning the contents cannot be changed. They are defined using parentheses and can be accessed by their index number.

- **Dictionaries:** Dictionaries are a collection of key-value pairs. They are defined using curly braces and items can be accessed by their key. Dictionaries are mutable.

- **Sets:** Sets are a collection of unique items. They are defined using curly braces and items cannot be accessed by their index number. Sets are mutable.

- **Strings:** Strings are a collection of characters. They are defined using single or double quotes and can be accessed by their index number. Strings are immutable.

Each data structure has its own advantages and disadvantages and is used in different scenarios. For example, lists are useful when you need to store a collection of items in a specific order and need to access them by their index number. Dictionaries are useful when you need to store data in a key-value format and need to access items by their key.

Task- 1 : Give the Difference between List, Tuple and set. Do Handson and put screenshots as per your understanding.

List: A list is a collection of items that are ordered and changeable. Lists are written with square brackets and items are separated by commas. Lists can contain any type of item, including other lists.

```
>>> #creating a list of fruits
>>> fruits =['apple', 'banana', 'orange']
>>>
>>> #Adding an item to list
>>> fruits.append('mango')
>>>
>>> #Removing an iteam from the list
>>> fruits.remove('banana')
>>>
>>> #Accessing an item from the list
>>> print(fruits[1])
orange
```

Tuple: A tuple is a collection of items that are ordered and unchangeable. Tuples are written with parentheses and items are separated by commas. Tuples can contain any type of item, including other tuples.

```
>>> #creating a tuple of colors
>>> colors = ('red', 'blue', 'green')
>>>
>>> #Trying to add an item to the tuple
>>> colors.append('yellow')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
>>>
>>> #Trying to remove an item from the tuple
>>> colors.remove('blue')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'remove'
>>>
>>> #Accessing an item from the tuple
>>> print(colors[1])
blue
```

Set: A set is a collection of items that are unordered and unchangeable. Sets are written with curly braces and items are separated by commas. Sets can contain any type of item, but each item can only appear once in a set.

```
>>> #Creating a set of numbers
>>> numbers = {1, 2, 3, 4, 5}
>>>
>>> #Adding an item to the set
>>> numbers.add(6)
>>>
>>> #Removing an item from the set
>>> numbers.remove(3)
>>>
>>> #Accessing an item from the set
>>> print(numbers)
{1, 2, 4, 5, 6}
```

Task -2 : Create below Dictionary and use Dictionary methods to print your favourite tool just by using the keys of the Dictionary.

```
fav_tools =
{
  1:"Linux",
  2:"Git",
  3:"Docker",
  4:"Kubernetes",
  5:"Terraform",
  6:"Ansible",
  7:"Chef"
}
```

```
>>> fav_tools ={
... 1: "Linux" ,
... 2: "Git" ,
... 3: "Docker" ,
... 4: "Kubernetes" ,
... 5: "Terraform" ,
... 6: "Ansible" ,
... 7: "Chef"
... }
>>> print(fav_tools[6])
Ansible
```

Task -3 : Create a List of cloud service providers eg.

cloud_providers = ["AWS","GCP","Azure"]
Write a program to add Digital Ocean to the list of cloud_providers and sort the list in alphabetical order.

```
>>> Cloud_providers = ["AWS", "AZURE", "GCP", "IBM", "ORACLE"]
>>> Cloud_providers.append("Digital Ocean")
>>> print("Sorting the list in ascending orders")
Sorting the list in ascending orders
>>> Cloud_providers.sort()
>>> print(C;oud_providers)
  File "<stdin>", line 1
    print(C;oud_providers)
           ^
SyntaxError: invalid syntax
>>> print(Cloud_providers)
['AWS', 'AZURE', 'Digital Ocean', 'GCP', 'IBM', 'ORACLE']
```