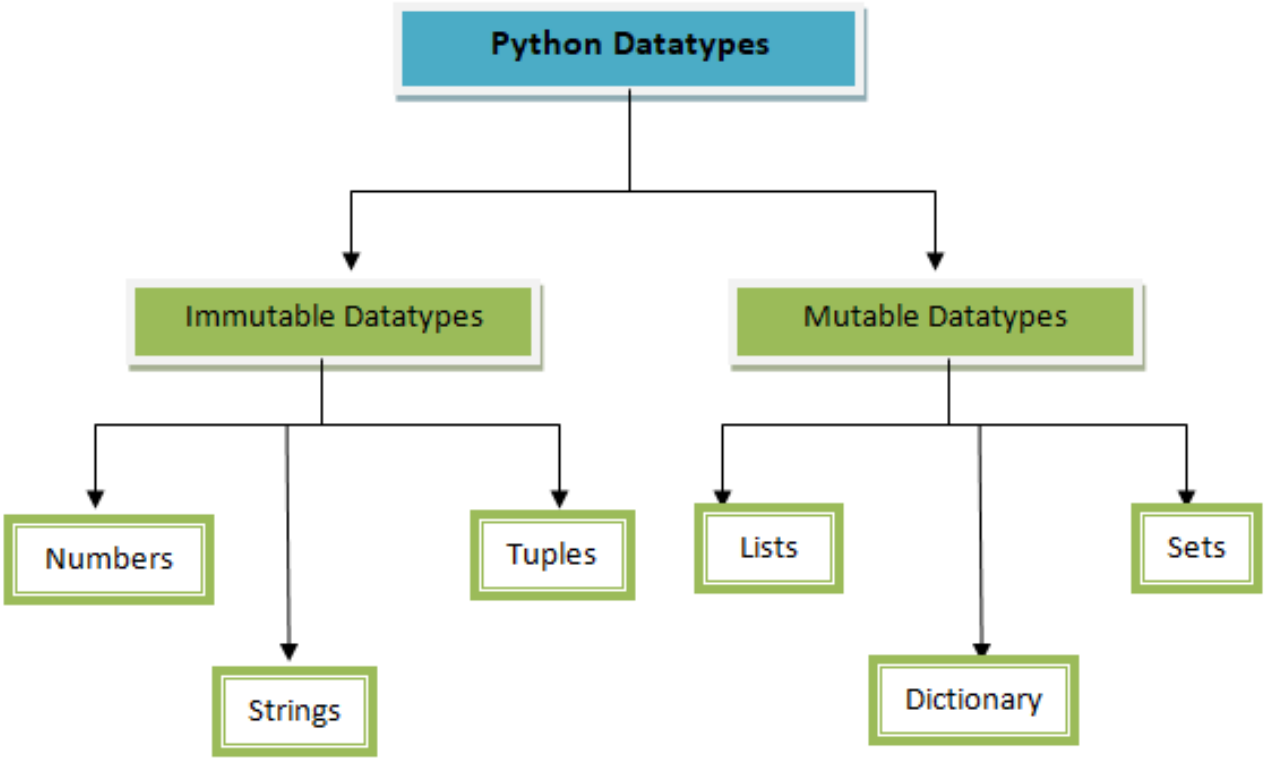
Reviving INDIA

Transforming Lives Digitally

Datatypes

-Python Series

Datatypes Classification



IMMutable Datatypes

- IMMutable data types are those data types whose value can't be changed after its creation.
- Whenever an object is created, it is assigned a unique object id.

Example : Nr Variable

```
number = 42
```

```
1657696608 #Output1
1657696640 #output2
print(id(number))

number += 1
print(id(number))
```

String

- text = **"Python Course"**
  print(id(text))

  text += **" with Reviving India"**
  print(id(text))

- 1565809038896
- 1565805802384

Python Object

- Every python object has some key elements in the structure

- An ID(Identity)
- A type
- A value

Example

- age = 62
- print(id(age))
- print(type(age))
- print(age)

What happens if someone wants to modify string?

- message = “Python Course“
- Print(message[0]) #’P’
- message[0] = 'p'
- print(message)

Error

- TypeError: 'str' object does not support item assignment

Tuple

- x= (10, 20, 30)
- print(x) #(10, 20, 30)
- Print(type(x)) #Class<Tuple>
- X[0] = 40
- print(x) #Error

Error - Tuple

```
Traceback (most recent call last):
  File "test.py", line 3, in < module >
    x[0] = 40
TypeError: 'tuple' object does not support
item assignment
```

Mutable Objects

- Mutable data types are those data types whose value can be changed after its creation.
- MUD operations can be done in Mutable objects
- During Iteration objects MUD is also achievable.

List

```
values = [4, 5, 6]
```

```
Id-2450343166664
Id-2450343166664
values2 = values
print(id(values))
print(id(values2))

Values[0]=8
print(values)   #[8, 5, 6]
print(values2)  #[8, 5, 6]
```

Tuple with List

```
skills = ["Programming",
"Machine Learning",
"Statistics"]
```

```
person = (129392130, skills)
print(type(person))
print(person)

skills[2] = "Maths"
print(person)
```

Continue..

<class 'tuple'>

(129392130, ['Programming', 'Machine Learning', 'Statistics'])

(129392130, ['Programming', 'Machine Learning', 'Maths'])

- If Immutable object contains only immutable objects, we cannot change their value.

```
unique_identifier = 42
age = 24
skills = ("Python", "pandas", "scikit-learn")
info = (unique_identifier, age, skills)
print(id(unique_identifier))
print(id(age))
print(info)
print(id(info))

unique_identifier = 50
age += 1
skills += ("machine learning", "deep learning")
info = (unique_identifier, age, skills)
print(id(unique_identifier))
print(id(age))
print(info)
print(id(info))
```

Points to Remember

- Concatenating string in loops wastes lots of **memory** , because strings are immutable, concatenating two strings together actually creates a third string which is the combination of the previous two.
- If you are iterating a lot and building a large string, you will waste a lot of memory creating and throwing away objects.

Continue…

- **Immutable** are quicker to access than mutable objects.
- Immutable objects are fundamentally expensive to "change", because doing so involves creating a copy.

- Changing **mutable** objects is cheap.