Python Programming

Data Types

Agenda :

- What is data type
- Numbers
- Sequence
- Set
- None
- Mappings

What is a data type ?

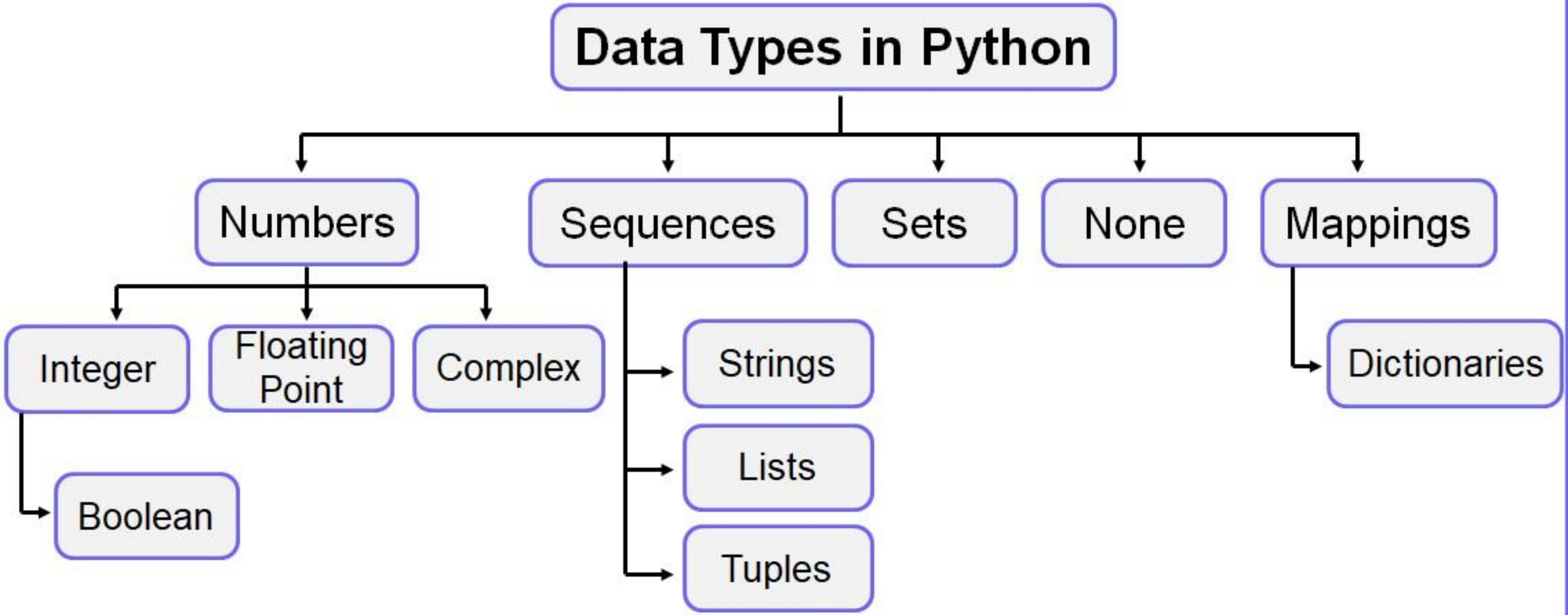
Data types are the classification or categorization of knowledge items.

It represents the kind of value that tells what operations can be performed on a particular data.

To define the values of various data types and check their data types we use the type() function.

teachoo

Data Types in Python



Numeric :

Number data type stores numerical values only. It is further classified into three different types: int, float and complex.

Type/ Class	Description	Examples
int	integer numbers	–12, –3, 0, 125, 2
float	real or floating point numbers	–2.04, 4.0, 14.23
complex	complex numbers	3 + 4j, 2 – 2j

Boolean data type (bool) is a subtype of integer. It is a unique data type, consisting of two constants, True and False. Boolean True value is non-zero, non-null and non-empty. Boolean False is the value zero.

Mutable and Immutable Data Types :

For certain data types, Python does not allow us to change the values once a variable of that type has been created and assigned values.

Variables whose values can be changed after they are created and assigned are called **mutable.**

Variables whose values cannot be changed after they are created and assigned are called **immutable.**

Mutables - List, Set, Dictionary

Immutables - Integer, float, boolean, complex, string, tuple

Sequence :

A Python sequence is an ordered collection of items, where each item is indexed by an integer. The three types of sequence data types available in Python -

1. Strings
2. Lists
3. Tuples

String :

- String is a group of characters. These characters may be alphabets, digits or special characters including spaces.
- String values are enclosed either in single quotation marks (e.g., ‘Hello’) or in double quotation marks (e.g., “Hello”).

```
>>> str1 = 'Hello Friend'

>>> str2 = "452”

>>> str3 = """Hello World!

                welcome to the world of Python""" (Multi Line String)
```

Accessing chars in String : The index specifies the character to be accessed in the string and is written in square brackets ([ ]). Index starts from 0 to n-1, where n is the length of string.

```
#initializes a string str1

>>> str1 = 'Hello World!'

#gives the first character of str1

>>> str1[0]

'H'
```

String Operations :

Concatenation :

```
>>> str1 = 'Hello'
>>> str2 = 'World!'
>>> str1 + str2
#Concatenated strings
'HelloWorld!'
```

Repetition :

```
>>> str1 = 'Hello'
>>> str1 * 2
'HelloHello'
```

Membership :

```
>>> str1 = 'Hello World!'
>>> 'W' in str1
True
```

Slicing :

- This can be done by specifying an index range.
- str1[n:m] returns all the characters starting from str1[n] till str1[m-1].

```
>>> str1 = 'Hello World!'

>>> str1[1:5]

'ello'
```

- If the first index is not mentioned, the slice starts from index 0.
- If the second index is not mentioned, the slicing is done till the length of the string.
- The slice operation can also take a third index that specifies the ‘step size’. For example, str1[n:m:k].

List :

List is a sequence of items separated by commas and the items are enclosed in square brackets [ ]

#To create a list >>>

list1 = [5, 3.4, "New Delhi", "20C", 45]

#print the elements of the list list1

```
>>> print(list1)
[5, 3.4, 'New Delhi', '20C', 45]
```

Accessing Elements in a List :

Same way as we do in strings.

Lists are **Mutable.**

List Operations:

Concatenation:

```
>>> list1 = [1,3,5,7,9] , >>> list2 = [2,4,6,8,10]

>>> list1 + list2

>>>[1, 3, 5, 7, 9, 2, 4, 6, 8, 10]
```

Repetition :

```
>>> list1 = ['Hello']
>>> list1 * 4
['Hello', 'Hello', 'Hello', 'Hello']
```

Membership:

```
>>> list1 = ['Red','Green','Blue']
>>> 'Green' in list1
True
```

Slicing:

```
>>> list1 =['Red','Green','Blue','Cyan', 'Magenta','Yellow','Black']
>>> list1[2:6]
['Blue', 'Cyan', 'Magenta', 'Yellow']
```

List Methods :

len() - Returns the length of the list

list() - Creates an empty list if no argument is passed.

append() - Appends a single element passed at the end of the list

extend() - Appends each element of the list to the end of the given list

insert() - Inserts an element at a particular index in the list

count() - Inserts an element at a particular index in the list

index() - Returns index of the first occurrence of the element in the list.

remove() - Removes the given element from the list.

pop() - Returns the element whose index is passed.

Other - reverse(), sort(), sorted(), min(), max(), sum()

Nested Lists :

When a list appears as an element of another list, it is called a nested list.

```
>>> list1 = [1,2,'a','c',[6,7,8],4,9]
```

To access the element of the nested list of list1, we have to specify two indices list1[i][j].

Copying Lists :

```
>>> list1 = [1,2,3]

list2 = list1

>>>list2

[1,2,3]
```

Tuple :

- A tuple is an ordered sequence of elements of different data types, such as integer, float, string, list or even a tuple enclosed in().
- This is unlike list, where values are enclosed in brackets [ ]. Once created, we **cannot change** the tuple.

```
#create a tuple

tuple1 >>> tuple1 = (10, 20, "Apple", 3.4, 'a')

#print the elements of the tuple tuple1

>>> print(tuple1)

(10, 20, "Apple", 3.4, 'a')
```

Accessing Elements in a Tuple :

```
>>> tuple1 = (2,4,6,8,10,12)

>>> tuple1[0]

2
```

- Tuple is **Immutable.**
- Concatenation, Repetition, Membership and Slicing works in tuple similar to lists.
- Tuples also can be nested.

Functions - len(), tuple(), count(), max(), min(), sum(), sorted()

Set :

- Set is an unordered collection of items separated by commas and the items are enclosed in curly brackets { }.
- A set is similar to list, except that it **cannot have duplicate** entries. Once created, elements of a set cannot be changed.

```
#create a set

>>> set1 = {10,20,3.14,"New Delhi"}

>>> print(set1)

{10, 20, 3.14, "New Delhi"}
```

None :

- None is a special data type with a single value.
- It is used to signify the absence of value in a situation.
- None supports no special operations, and it is neither same as False nor 0 (zero).

```
>>> myVar = None

>>> print(type(myVar))

<class ‘NoneType’>
```

Mapping:

Mapping is an unordered data type in Python. Currently, there is only one standard mapping data type in Python called dictionary.

Dictionary:

- Dictionary in Python holds data items in key-value pairs.
- Items in a dictionary are enclosed in curly brackets { }.
- Dictionaries permit faster access to data.
- Every key is separated from its value using a colon (:) sign.

- The key : value pairs of a dictionary can be accessed using the key.
- The keys are usually strings and their values can be any data type.
- In order to access any value in the dictionary, we have to specify its key in square brackets [ ].

```
#create a dictionary

>>> dict1 = {'Fruit':'Apple', 'Climate':'Cold', 'Price(kg)':120}

>>> print(dict1)

{'Fruit': 'Apple', 'Climate': 'Cold', 'Price(kg)': 120}

## accessing values

>>> print(dict1['Price(kg)'])

120
```

Dictionaries are **Mutable**.

Adding a new item -

```
>>> dict1['Meena'] = 78
```

Modifying an Existing Item -

```
 >>>dict1['Suhel'] = 93.5
```

Membership :

Key in dict:

key, value in dict1.items():

Methods :

len(), dict(), keys(), values(), items(), get()

update() - appends the key-value pair of the dictionary.

del() - Deletes the item with the given key.

To delete the dictionary from the memory we write: del Dict_name

clear() - clear all the items of the dictionary

Nested Dictionaries :

Nesting Dictionary means putting a dictionary inside another dictionary. Nesting is of great use as the kind of information we can model in programs is expanded greatly.

Eg -

```
Dict = { 'Dict1': {'name': 'Ali', 'age': '19'},
        'Dict2': {'name': 'Bob', 'age': '25'}}
## access - Dict[“Dict1”][‘name’]
>>> “Ali”
```

References and Links :

Python Documentation : https://docs.python.org/3/

Jupyter Notebook : https://jupyter-notebook.readthedocs.io/en/stable/

Github : https://github.com/lunatic-bot/PythonTraining

Thank You