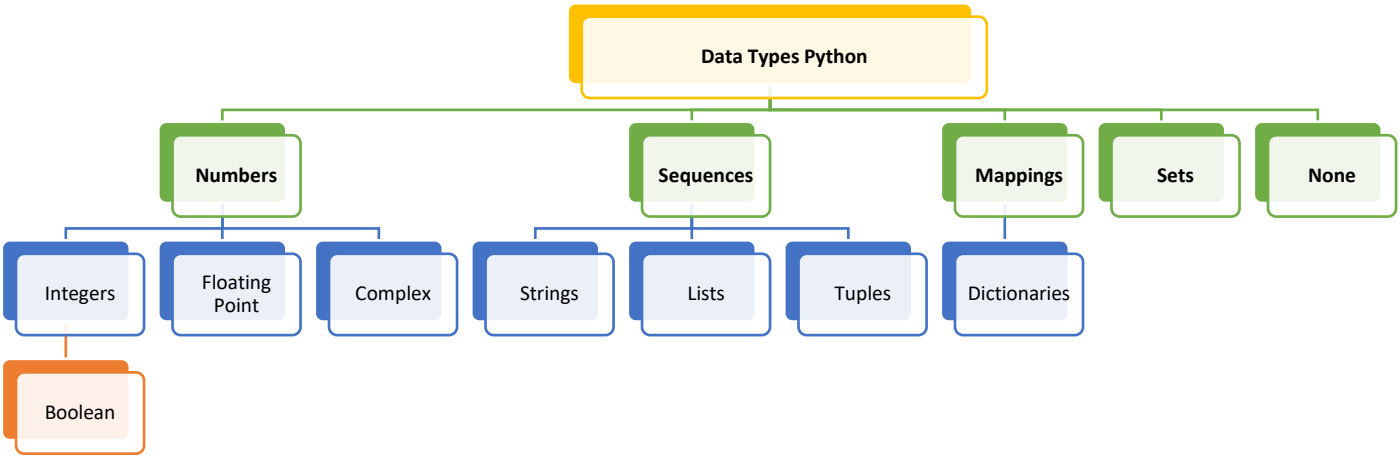
Data types

- Variables can hold different types of values and every value has a data-type.
- The Data Type defines the type of data values a variable can hold and the operations that can be performed on that data.
- To get the data type of any variable the type() function is used:

```
#Print the data type of the variable x
>>>x                                                    = 5
>>>print(type(x))
```

Output = <class 'int'>

Built-in Data Types



1] Numbers

- Number data type stores numerical values only.
- It is further classified into three different types:

a) Int b) Float c) 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

a) Int

Int, or integer, is a whole number, positive or negative and numbers without decimals. Python has no restriction on the length of an integer.

Example:

```
>>>x = 1
>>>y = 35656222554887711
>>>z = -3255522

>>>print(type(x))
>>>print(type(y))
>>>print(type(z))
```

Output =

```
<class 'int'>
<class 'int'>
<class 'int'>
```

b) Float

Float, or "floating point number" is a number, positive or negative, containing one or more decimals.. It is accurate upto 15 decimal points. Example:

```
>>>x = 1.10
>>>y = 1.0
>>>z = -35.59

>>>print(type(x))
>>>print(type(y))
>>>print(type(z))
```

Output =

```
<class 'float'>
<class 'float'>
<class 'float'>
```

c) Complex

A complex number contains an ordered pair, i.e., x + iy where x and y denote the real and imaginary parts, respectively. Complex numbers are written with a "j" as the imaginary part. Example:

```
>>>x = 3+5j
>>>y = 5j
>>>z = -5j

>>>print(type(x))
>>>print(type(y))
>>>print(type(z))
```

Output =

```
<class 'complex'>
<class 'complex'>
<class 'complex'>
```

Note:

Boolean

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

2] Sequence

a) String

- A string is a **group of characters** and can include **alphabets, digits or special characters** including spaces.
- **We can use single, double, or triple quotes to define a string.**
- String values are **enclosed either in single quotation marks or in double quotation**. For Example: String written as 'hello' is the same as "hello"
- The **quotation marks are not a part of the string**, they are used to mark the beginning and end of the string for the interpreter.
- We **cannot perform numerical operations on strings**, even when the string contains a numeric value.

```
>>>str1 = 'Hello Friend'
>>>str2 = "452"
#We can display a string with the print() function:
>>>print("Hello")
>>>print('Hello')
```

b) List

- List can contain data of different types.
- Items stored in the list are **separated by commas and are enclosed in square brackets [ ].**

```
#To create a list
>>> list1 = [5, 3.4, "Python For Data Science", "20C", 45]
#print the elements of the list list1
>>> print(list1)
```

Output = [5, 3.4, 'Data Science', '20C', 45]

c) Tuples

- Tuple also can contain data of different types.
- Items stored in the tuple are **separated by commas and are enclosed in parenthesis ( ).**
- A tuple is a read-only data structure as **we can't modify the value of the items of a tuple.** However we can add and delete values in a tuple.

```
#create a tuple tuple1
>>> tuple1 = (10, 20, "Data Science", 3.4, 'a')
#print the elements of the tuple tuple1
>>> print(tuple1)
```

Output = (10, 20, " Data Science", 3.4, 'a')

3) Set

- **Set is an unordered collection of items** that can contain values of different data types.
- Items are **separated by commas and are enclosed in curly brackets { }.**
- A set is similar to list, except that **it cannot have duplicate entries.**

```
#create a set
>>> set1 = {10, 20, 3.14," Data Science "}
>>> print(type(set1))
Output = <class 'set'>
>>> print(set1)
Output = {10, 20, 3.14, "Data Science"}
#duplicate elements are not included in set
>>> set2 = {1,2,1,3}
>>> print (set2)
Output = {1, 2, 3}
```

4) 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 (0).

```
>>> a = None
>>> print(type(a))
Output = <class 'NoneType'>
>>> print(a)
Output = None
>>> a = “Data Science”
>>> print(a)
Output = Data Science
```

5) Mapping

- Currently, there is only one standard mapping data type in Python called dictionary.

a) Dictionary

- Dictionary is an ordered set (from Python 3.7 and above versions) of **data items in key-value pairs.**
- Items in a dictionary are enclosed in **curly brackets { }.**
- Every **key is separated from its value using a colon (:) sign.**
- Each key holds a single value and the **key : value pairs** of a dictionary **can be accessed using the key. in square brackets [ ].**
- Dictionaries **permits faster access to data.**

```
#create a dictionary
>>> dict1 = {'Fruit':'Apple', 'Climate':'Cold', 'Price(kg)':120}
>>> print(dict1)
Output = {'Fruit': 'Apple', 'Climate': 'Cold', 'Price(kg)': 120}
>>> print(dict1['Price(kg)'])
Output = 120
```

Note: 1

Ordered means that the items have a defined order, and that order will not change. If you add new items, the new items will be placed at the end of the list.

Note: 2

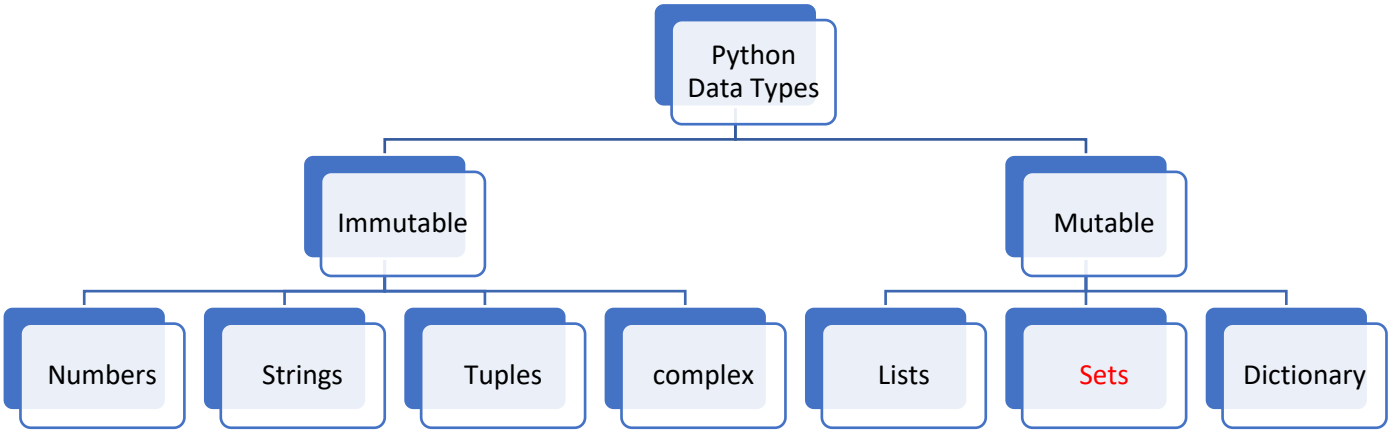
Mutable Data Type

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

Immutable Data Type

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

Python Data Types are classified into mutable and immutable as:



Note: 3

- Set items are unchangeable, but we can add or remove items whenever you like.

Note: 4

Usage of Data Type (Which Data Type should be when?)

✓ Lists are used when we need a simple iterable collection of data that may go for frequent modifications.
For example, if we store the names of students of a class in a list, then it is easy to update the list when some new students join or some leave the course.

✓ Tuples are used when we do not need any change in the data.
For example, names of months in a year.

✓ Sets are used when we need uniqueness of elements and to avoid duplicacy it is preferable to use sets.
For example, list of items in a museum.

✓ Dictionaries are used if our data is being constantly modified or we need a fast lookup based on a custom key or we need a association between the key : value pair.
For Example, A mobile phone book is a good application of dictionary.

Assigning Data Types to a variable:

The data type is set when you assign a value to a variable. This can be done as:

Example	Data Type
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = {"name" : "John", "age" : 36}	dict
x = {"apple", "banana", "cherry"}	set
x = None	NoneType
x = True	bool

ExplicitIt setting the data type of a variable:

To specify the specific data type:

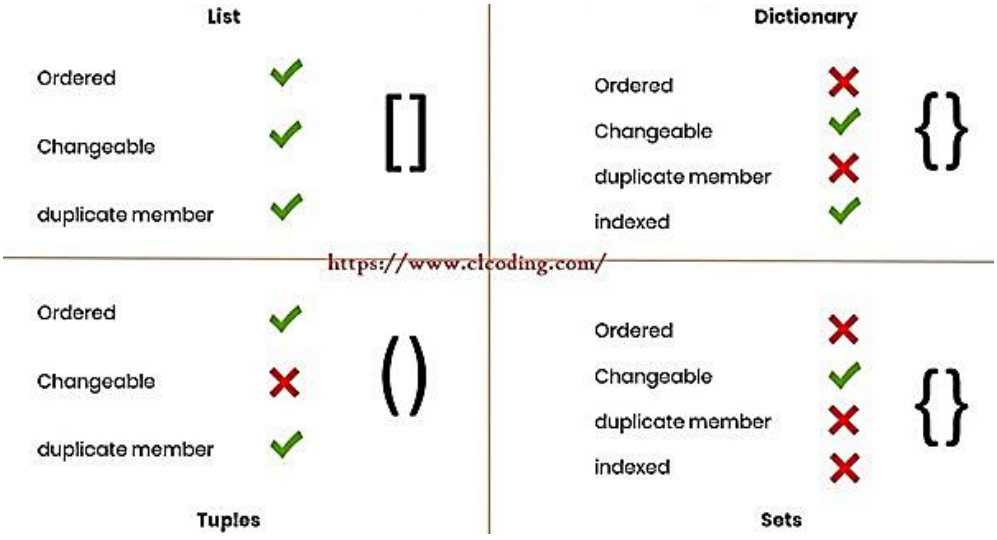
Example	Data Type
x = str("Hello World")	str
x = int(20)	int
x = float(20.5)	float
x = complex(1j)	complex
x = list(("apple", "banana", "cherry"))	list
x = tuple(("apple", "banana", "cherry"))	tuple
x = dict(name="John", age=36)	dict
x = set(("apple", "banana", "cherry"))	set
x = bool(5)	bool

Additional Points:

List, Tuple, Set, and Dictionary are the data structures in python that are used to store and organize the data in an efficient manner and the differences between them are:

List	Tuple	Set	Dictionary
List is a non-homogeneous data structure that stores the elements in single row and multiple rows and columns	Tuple is also a non-homogeneous data structure that stores single row and multiple rows and columns	Set data structure is also non-homogeneous data structure but stores in single row	Dictionary is also a non-homogeneous data structure which stores key value pairs
List can be represented by [ ]	Tuple can be represented by ( )	Set can be represented by { }	Dictionary can be represented by { }
List allows duplicate elements	Tuple allows duplicate elements	Set will not allow duplicate elements	dictionary doesn't allow duplicate keys.
List can use nested among all	Tuple can use nested among all	Set can use nested among all	Dictionary can use nested among all
Example: [1, 2, 3, 4, 5]	Example: (1, 2, 3, 4, 5)	Example: {1, 2, 3, 4, 5}	Example: {1, 2, 3, 4, 5}
List can be created using **list()** function	Tuple can be created using **tuple()** function.	Set can be created using **set()** function	Dictionary can be created using **dict()** function.
List is mutable i.e we can make any changes in list.	Tuple is immutable i.e we can not make any changes in tuple	Set is mutable i.e we can make any changes in set. But elements are not duplicated.	Dictionary is mutable. But Keys are not duplicated.
List is ordered	Tuple is ordered	Set is unordered	Dictionary is ordered (Python 3.7 and above)
Creating an empty list l=[]	Creating an empty Tuple t=()	Creating a set a=set() b=set(a)	Creating an empty dictionary d={}

Remembering the differences:



Applications of List, Set, Tuple, and Dictionary

List:

- Used in JSON format
- Useful for Array operations
- Used in Databases

Tuple:

- Used to insert records in the database through SQL query at a time.Ex: (1.’sravan’, 34).(2.’geek’, 35)
- Used in parentheses checker

Set:

- Finding unique elements
- Join operations

Dictionary:

- Used to create a data frame with lists
- Used in JSON