Standard Datatypes

In [ ]:
```
Standard Datatype that are common to every programming Language --> Integer,Float, String, Boolean
```

TypeCasting -- Conversion of One Datatype to Another

In [ ]:
```
For Converting Any Datatype to int we have --> int()
For Converting Any Datatype to float we have --> float()
For Converting Any Datatype to complex we have --> complex()
For Converting Any Datatype to string we have --> str()
For Converting Any Datatype to Boolean we have --> bool()
```

Possibilities of Conversions of One Datatype to Another

Conversion of Float to Int datatype -- Possible

In [82]:
```
x=123.234
y=int(x)
print(type(y))
```

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

Conversion of complex datatype into integer datatype -- Not Possible

In [83]:
```
x=1+2j
y=int(x)
print(y)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [83], in <cell line: 2>()
      1 x=1+2j
----> 2 y=int(x)
      3 print(y)

TypeError: can't convert complex to int
```

In [84]:
```
x1=0j
y=int(x)
print(y)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [84], in <cell line: 2>()
      1 x=1+0j
----> 2 y=int(x)
      3 print(y)

TypeError: can't convert complex to int
```

Conversion of Boolean datatype to integer datatype -- Possible

In [85]:
```
x=True
y=int(x)
print(y)
```

```
1
```

Conversion of String Datatype to integer datatype -- Possible/Not Possible

In [86]:
```
x="10"
y=int(x)
print(y)
```

```
10
```

In [ ]:
```
Note: If we are converting string datatype to integer datatype then it is mandatory that within the
      quotes the literal/data is in the form of Integer only.
x="10.5"
y=int(x)
print(y) #Error

x="abc"
y=int(x)
print(y) #Error
```

Conversion of Integer Datatype to Float datatype -- Possible

In [87]:
```
x=10
y=float(x)
print(y)
```

```
10.0
```

Conversion of Complex Datatype to Float datatype -- Not Possible

In [88]:
```
x=10+20j
y=float(x)
print(y)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [88], in <cell line: 2>()
      1 x=10+20j
----> 2 y=float(x)
      3 print(y)

TypeError: can't convert complex to float
```

Conversion of Boolean Datatype to Float datatype -- Possible

In [89]:
```
x=True
y=float(x)
print(y)
```

```
1.0
```

Conversion of String Datatype to Float datatype -- Not Possible/Possible

In [90]:
```
x="10.5"
y=float(x)
print(y)
```

```
10.5
```

In [91]:
```
x="23"
y=float(x)
print(y)
```

```
23.0
```

In [ ]:
```
Note: If we are converting string datatype to Float datatype then it is mandatory that within the
      quotes the literal/data is in the form of Float only/Integer. Then only typecasting will be done

x="ten.5"
y=float(x)
print(y) #Error
```

Conversion of Integer Datatype to Complex datatype -- Possible

In [29]:
```
x=10
y=complex(x)
print(y)
```

```
(10+0j)
```

Conversion of Float Datatype to Complex datatype -- Possible

In [30]:
```
x=10.5
y=complex(x)
print(y)
```

```
(10.5+0j)
```

Conversion of Bool datatype to complex datatype -- Possible

In [92]:
```
x=False
y=complex(x)
print(y)
```

```
0j
```

Conversion of String datatype to complex datatype -- Possible/Not Possible

In [93]:
```
x="10+20j"
y=complex(x)
print(y)
x="10"
y=complex(x)
print(y)
x="10.5"
y=complex(x)
print(y)
```

```
(10+20j)
(10+0j)
(10.5+0j)
```

In [ ]:
```
Note: If we are converting string datatype to Complex datatype then it is mandatory that within the
      quotes the literal/data is in the form of Float/Integer/complex.
      Then only typecasting will be done
x="ten.5"
y=complex(x)
print(y) #Error
```

Boolean Conversion -- You can Convert Any Datatype to Boolean --Possible

In [ ]:
```
0 and " "--> False
Other than this answer is True
```

In [94]:
```
x=bool(0.0)
x
```

Out[94]:
```
False
```

In [95]:
```
x=bool("")
x
```

Out[95]:
```
False
```

In [96]:
```
x=bool(0+0j)
x
```

Out[96]:
```
False
```

In [97]:
```
x=bool(-2)
x
```

Out[97]:
```
True
```

In [98]:
```
x=bool("String")
x
```

Out[98]:
```
True
```

In [99]:
```
x=bool(2-5)
x
```

Out[99]:
```
True
```

In [72]:
```
x=bool(1.5)
x
```

Out[72]:
```
True
```

String Conversion -- You can convert any datatype to string -- Possible

In [51]:
```
x=10
y=str(x)
print(type(y))
```

```
<class 'str'>
```

In [100..
```
x=10.5
y=str(x)
print(y)
```

```
10.5
```

In [101..
```
x=10.5+20j
y=str(x)
print(y)
```

```
(10.5+20j)
```

In [102..
```
x=True
y=str(x)
print(y)
```

```
True
```

In [ ]:
```
Mutable means we can change the content of it
Immutable means we cannot make any change in it.
```

List Datatype

In [ ]:
```
-->List is a collection of Dissimilar Elements.
-->If we want to Store multiple Elements as a single entity then we can use List .
-->Square Brackets are used to represent List Datatype.
-->List Datatype is mutable.(Means we can change the Content/Element of the List).
```

In [105..
```
x=[10,20,30,40,50]
x[1]="Hello World"
print(x)
print(type(x))
```

```
[10, 'Hello World', 30, 40, 50]
<class 'list'>
```

Tuple Datatype

In [ ]:
```
-->Tuple is a collection of Dissimilar Elements.
-->If we want to Store multiple Elements as a single entity
   then we can use Tuple .
-->Parenthesis Brackets are used to represent Tuple Datatype.
-->Tuple Datatype is immutable.(Means we cannot change the Content/Element of the Tuple).
-->It is read only mode of List.
```

In [ ]:
```
x=(10,20,30,40,50)
x[0]="Hello World"  #tuples are immutable we cannot change the content.
print(x)
```

Set Datatype

In [ ]:
```
--> Set is also a collection of Element.
--> In Sets Duplicate values are not allowed.
--> Curly Braces are used to represent Sets.
--> Set are mutable(Means we can change the content/element of Set)
```

In [107..
```
x={1,2,3,4,5,7,8,8,8,8,7,7,7,6,6,6,5,5,5,4,4}
print(type(x))
print(x)
```

```
<class 'set'>
{1, 2, 3, 4, 5, 6, 7, 8}
```

Dictionary Datatype

In [ ]:
```
--> Dictionary is also a collection of Element That will store elements in the form of Keys and Value
--> Curly Braces are Used to Represent Dictionary
--> Dictionary keys are unique They must not be duplicate(If you use duplicate keys then only last one
                                                          will be considered)
--> Dictionary are also mutable.(You can change the content of Dictionary as well)
```

In [110..
```
x={"name":"Pratyush","Class":"M.tech","name":"Taskeen"}
x["name"]
```

Out[110]:
```
'Taskeen'
```

Summary:

In [ ]:
```
Lists are mutable --> we can change the content of the list
Tuples are immutable --> we cannot change the content of the tuple
Set are mutable and Duplicates not allowed --> we can change the content of the set but duplicates
are not allowed
Dictionary are mutable --> we can change the keys and values in dictionary. Duplicate keys are not
allowed but values may be duplicate
```

Conversions:

In [ ]:
```
Conversions:
    integer to float --> possible
    integer to boolean --> possible
    integer to string --> Possible if within quotes data is in integer format else not
    float to int --> possible
    float to boolean --> possible
    float to string -->Possible if within quotes data is in integer/float format else not
    boolean --> 0 and "" are false rest all are True
    string --> you can convert any data to string
```

Two Important Functions

In [ ]:
```
print() --> Display the output
input() --> taking the input from the user
```

Use Case of TypeCasting

In [111..
```
x=input("Enter number")
z=int(x)
y=input("Enter number")
print(y)
print(z+z)
```

```
Enter number10
Enter number10
20
```