Fundamental Category Data-type

The Main purpose of fundamental category data-type is that "to store single value".

The fundamental category contains 4 Data-types

1) Int

2) Float

3) Bool

4) Complex

In [ ]:

1) Int:

Int is one of the pre-defined class and it is treated as fundamental data-type.

The main purpose of 'int' Data-type is that "to store integer/whole/integral" values.

Integer/whole/integral values are nothing but without decimal places.

In [2]:
```
a = 123 # Here a = var and 123 = literals(values)
print(a,type(a),id(a))
```

```
123 <class 'int'> 140718899546360
```

With 'int' DT, we can also store number system Data.

In any programming language(Python,C,C++,Java), we have 4-types of number systems.

They are:

1) Decimal number system(Default)

It is one of the default number system.

Digits: 0 1 2 3 4 5 6 7 8 9

Total No.of digits: 10

Base: 10

2) Binary number system

The digits in binary number systems are

Digits: 0 1

Total No.of Digits: 2

Base: 2

To store binary data in python program environment, binary data must be proceed with 0b (or) 0B

Example: varname: ob Binary Data

varname: ob Binary Data

Even we store binary data and when we display the binary data converted into decimal number system.

In [3]:
```
a = 0b1110
print(a,type(a),id(a))
```

```
14 <class 'int'> 140718899542872
```

In [4]:
```
b = 0B1110
print(b,type(b),id(b))
```

```
14 <class 'int'> 140718899542872
```

In [5]:
```
c = 0b10102 # Syntax error: Invalid digit 2
print(c,type(c),id(c))
```

```
  Cell In[5], line 1
    c = 0b10102 # Syntax error: Invalid digit 2
                ^
SyntaxError: invalid digit '2' in binary literal
```

In [ ]:

3) Octal number system

The digits in octal number system.

Digits: 0 1 2 3 4 5 6 7

Total no.of digits: 8

Base: 8

Syntax: varname: 0o octal data

varname: 0O octal data

Even we store octal data and we display the octal data converted into decimal number.

In [ ]:

4) Hexa decimal number system

The digits in hexa decimal number system are

Digits: 0 1 2 3 4 5 6 7 8 9 A(10) B(11) C(12) D(13) E(14) F(15)

Total no.of digits: 16

Base: 16

In [6]:
```
a = 0xAC
print(a,type(a),id(a))
```

```
172 <class 'int'> 140718899547928
```

In [7]:
```
a = 0xBEE
print(a,type(a),id(a))
```

```
3054 <class 'int'> 1601274443632
```

In [8]:
```
a = 0xFACE
print(a,type(a),id(a))
```

```
64206 <class 'int'> 1601274443760
```

In [9]:
```
a = 0x300
print(a,type(a),id(a))
```

```
768 <class 'int'> 1601274445456
```

In [10]:
```
a = 0xFACER # SyntaxError: invalid hexadecimal literal
print(a,type(a))
```

```
  Cell In[10], line 1
    a = 0xFACER # syntax error - invalid hexa decimal
              ^
SyntaxError: invalid hexadecimal literal
```

In [ ]:
```

```

Base Conversion techniques

The purpose of base conversion technique in python is that "converting one base value to another base value".

We have 3 base conversion techniques. they are:

a) bin()

b) oct()

c) hex()

bin()

This function is used for converting other number system data into binary number system.

syntax: varname = bin(dec/hex/oct)

In [12]:
```
a = 15
b = bin(a)
print(b,type(b),id(b))
```

```
0b1111 <class 'str'> 1601268607968
```

In [13]:
```
a = 25
b = bin(a)
print(b,type(b),id(b))
```

```
0b11001 <class 'str'> 1601269948336
```

In [14]:
```
a = 0o22 # octal
b = bin(a)
print(b,type(b))
```

```
0b10010 <class 'str'>
```

In [15]:
```
a = 0xf # hexa
b = bin(a)
print(b,type(b))
```

```
0b1111 <class 'str'>
```

In [16]:
```
a = 0xface # hexa
b = bin(a)
print(b,type(b))
```

```
0b1111101011001110 <class 'str'>
```

In [ ]:
```

```

oct()

This function is used for converting other number system into octal.

In [18]:
```
a = 18
b = oct(a)
print(b,type(b),id(b))
```

```
0o22 <class 'str'> 1601268609936
```

In [19]:
```
c = oct(234)
print(c,type(c),id(c))
```

```
0o352 <class 'str'> 1601268608784
```

In [ ]:
```

```

hex()

This function is used for converting other number system into hexa decimal number system.

In [20]:
```
a = 15
b = hex(a)
print(b,type(b),id(b))
```

```
0xf <class 'str'> 1601268610560
```

In [21]:
```
b = hex(123)
print(b,type(b))
```

```
0x7b <class 'str'>
```

In [22]:
```
print(hex(172))
```

```
0xac
```

In [ ]:

2) Float

Flaot is one of the pre-defined class and treated as fundamental DT.

The purpose of float DT is that "To store floating point value" (or) "Read constant values"(Number with decimal places).

Example: 13.85, 54.9, 89.8 and -34.6 -98.3 -67.3

In this 13 is called integer and 0.85 is called decimal part.

In [24]:
```
a = 13.85
print(a,type(a))
```

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

In [25]:
```
b = 54.9
print(b,type(b))
```

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

In [26]:
```
c = -34.6
print(c,type(c))
```

```
-34.6 <class 'float'>
```

In [27]:
```
d = -98.3
print(d,type(d))
```

```
-98.3 <class 'float'>
```

In [ ]:

bool

'bool' is one of the pre-defined class and treated as fundamental data type.

The main purpose if bool DT is that "to store True or False"(logical values).

Internally T = 1 and F = 0

In [28]:
```
a = True
print(a,type(a))
```

```
True <class 'bool'>
```

In [29]:
```
b = False
print(b,type(b))
```

```
False <class 'bool'>
```

In [30]:
```
print(True + True)
```

```
2
```

In [31]:
```
print(True * True)
```

```
1
```

In [32]:
```
print(True + False)
```

```
1
```

In [33]:
```
print(True * False)
```

```
0
```

In [34]:
```
print(2 * True + False)
```

```
2
```

In [35]:
```
print(True + 0b1111)
```

```
16
```

In [36]:
```
print(True + 0xF)
print(True / True)
print(True // True)
```

```
16
1.0
1
```

In [ ]:

Complex

The purpose of complex DT is that "to store imaginary data / complex numbers".

The General formal of complex numbers is

a + bj or a - bj

Here a = real part

b = imaginary part

j = sqrt(-1)

To get real and imaginary part from complex object we use two pre-defined attributes.

They are:

REAL

IMAGINARY

In [38]:
```
a = 3 + 4j
print(a,type(a))
```

```
(3+4j) <class 'complex'>
```

In [39]:
```
a = 3 - 4j
print(a,type(a))
```

```
(3-4j) <class 'complex'>
```

In [40]:
```
a = 2.4 + 4.5j
print(a,type(a))
```

```
(2.4+4.5j) <class 'complex'>
```

In [41]:
```
a = 10 + 40j
print(a,type(a))
print(a.real)
print(a.imag)
```

```
(10+40j) <class 'complex'>
10.0
40.0
```

In [42]:
```
a = 14 + 53j
print(a,type(a))
print(a.real)
print(a.imag)
```

```
(14+53j) <class 'complex'>
14.0
53.0
```

In [43]:
```
a = 35 + 4.9j
print(a,type(a))
print(a.real)
print(a.imag)
```

```
(35+4.9j) <class 'complex'>
35.0
4.9
```

In [44]:
```
a = 2 + 3j
b = 4 + 5j
print(a,type(a))
print(b,type(b))
print("----------------------")
c = a + b
print(c,type(c))
```

```
(2+3j) <class 'complex'>
(4+5j) <class 'complex'>
----------------------
(6+8j) <class 'complex'>
```

In [45]:
```
a = 8 + 3j
b = 3 + 5j
print(a,type(a))
print(b,type(b))
print("----------------------")
c = a + b
print(c,type(c))
```

```
(8+3j) <class 'complex'>
(3+5j) <class 'complex'>
----------------------
(11+8j) <class 'complex'>
```

In [46]:
```
a = True + 2j
print(a,type(a))
```

```
(1+2j) <class 'complex'>
```

In [47]:
```
a = 0b1010 -5j
print(a,type(a))
```

```
(10-5j) <class 'complex'>
```

In [ ]:
```

```