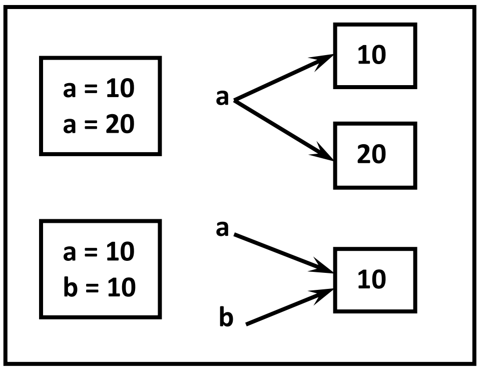
DATA TYPES

- **Data Type represents the type of data present inside a variable.**
- **In Python we are not required to specify the type explicitly. Based on value provided, the type will be assigned automatically.Hence Python is dynamically Typed Language.**

Python contains the following inbuilt data types

1) **Int**
2) **Float**
3) **Complex**
4) **Bool**
5) **Str**
6) **Bytes**
7) **Bytearray**
8) **Range**
9) **List**
10) **Tuple**
11) **Set**
12) **Frozenset**
13) **Dict**
14) **None**



<u>Note:</u> Python contains several inbuilt functions

1) <u>type()</u>

to check the type of variable

2) <u>id()</u>

to get address of object

3) print()

to print the value

In Python everything is an Object.

1) int Data Type:

We can use int data type to represent whole numbers (integral values)

Eg: a = 10
    type(a) #int

Note:

- In Python2 we have long data type to represent very large integral values.
- But in Python3 there is no long type explicitly and we can represent long values also by using int type only.

We can represent int values in the following ways

1) Decimal form
2) Binary form
3) Octal form
4) Hexa decimal form

I) Decimal Form (Base-10):

- It is the default number system in Python
- The allowed digits are: 0 to 9
- Eg: a =10

II) Binary Form (Base-2):

- The allowed digits are : 0 & 1
- Literal value should be prefixed with 0b or 0B

- Eg: a = 0B1111
  a = 0B123
  a = b111

III) Octal Form (Base-8):

- The allowed digits are : 0 to 7
- Literal value should be prefixed with 0o or 0O.

- **Eg:** a = 0o123
  a = 0o786

IV) Hexa Decimal Form (Base-16):

- The allowed digits are: 0 to 9, a-f (both lower and upper cases are allowed)
- Literal value should be prefixed with 0x or 0X

- **Eg:** a = 0XFACE
  a = 0XBeef
  a = 0XBeer

Note: Being a programmer we can specify literal values in decimal, binary, octal and hexa decimal forms. But PVM will always provide values only in decimal form.

- a=10
- b=0o10
- c=0X10
- d=0B10
- print(a)10
- print(b)8
- print(c)16
- print(d)2

Base Conversions

Python provide the following in-built functions for base conversions

1) bin():

We can use bin() to convert from any base to binary

```
>>> bin(15)
'0b1111'
>>> bin(0o11)
'0b1001'
>>> bin(0X10)
'0b10000'
```

2) oct():

We can use oct() to convert from any base to octal

```
>>> oct(10)
'0o12'
>>> oct(0B1111)
'0o17'
>>> oct(0X123)
'0o443'
```

3) hex():

We can use hex() to convert from any base to hexa decimal

```
>>> hex(100)
'0x64'
>>> hex(0B111111)
'0x3f'
>>> hex(0o12345)
'0x14e5'
```

2) Float Data Type:

- We can use float data type to represent floating point values (decimal values)

  Eg: f = 1.234

  type(f)   float

- We can also represent floating point values by using exponential form (Scientific Notation)

  Eg: f = 1.2e3 → instead of 'e' we can use 'E'

  print(f)   1200.0

- The main advantage of exponential form is we can represent big values in less memory.

***Note:**

We can represent int values in decimal, binary, octal and hexa decimal forms. But we can represent float values only by using decimal form.

```
 >>> f=0B11.01
  File "<stdin>", line 1
    f=0B11.01
         ^
SyntaxError: invalid syntax

>>> f=0o123.456
 SyntaxError: invalid syntax

>>> f=0X123.456
 SyntaxError: invalid syntax
```

3) Complex Data Type:

- A complex number is of the form

$$\begin{tikzcd}
 & a + bj \arrow[dl] \arrow[d] \arrow[r] & \begin{array}{l} j2 = -1 \\ j = \sqrt{-1} \end{array} \\
\text{Real Part} & \text{Imaginary Part} &
\end{tikzcd}$$

- 'a' and 'b' contain Intergers OR Floating Point Values.

Eg: 3 + 5j
10 + 5.5j
0.5 + 0.1j

- In the real part if we use int value then we can specify that either by decimal, octal, binary or hexa decimal form.
- But imaginary part should be specified only by using decimal form.

```
>>> a=0B11+5j
>>> a
(3+5j)
>>> a=3+0B11j
  SyntaxError: invalid syntax
```

- Even we can perform operations on complex type values.

```
>>> a=10+1.5j
>>> b=20+2.5j
>>> c=a+b
>>> print(c)
(30+4j)
```

```
>>> type(c)
<class 'complex'>
```

Note: Complex data type has some inbuilt attributes to retrieve the real part and imaginary part

c = 10.5+3.6j

c.real → 10.5
c.imag → 3.6

We can use complex type generally in scientific Applications and electrical engineering Applications.

4) bool Data Type:

- We can use this data type to represent boolean values.
- The only allowed values for this data type are:
- True and False

- Internally Python represents True as 1 and False as 0

    b = True
    type(b) →bool

Eg:
a = 10
b = 20
c = a<b
print(c) → True

True+True → 2
True-False → 1

5) str Data Type:

- str represents String data type.
- A String is a sequence of characters enclosed within single quotes or double quotes.

- s1='ravi'
- s1=" ravi"

- **By using single quotes or double quotes we cannot represent multi line string literals.**

- **s1="frame
  work"**

- **For this requirement we should go for triple single quotes(''') or triple double quotes(""")**

- **s1='''frame
  work'''**

- **s1="""frame
  work"""**

- **We can also use triple quotes to use single quote or double quote in our String.**
- **''' This is " character'''
  ' This i " Character '**

- **We can embed one string in another string**
- **'''This "Python class very helpful" for java students'''**

Slicing of Strings:

1) **slice means a piece**
2) **[ ] operator is called slice operator, which can be used to retrieve parts of String.**
3) **In Python Strings follows zero based index.**
4) **The index can be either +ve or -ve.**
5) **+ve index means forward direction from Left to Right**
6) **-ve index means backward direction from Right to Left**

-5	-4	-3	-2	-1
b	a	s	i	c
0	1	2	3	4

```
>>> s="basic"
>>> s[0]
'b'
>>> s[1]
'a'
>>> s[-1]
'c'
>>> s[40]
```

IndexError: string index out of range

```
>>> s[1:40]
'asic'
>>> s[1:]
'asic'
>>> s[:4]
'basi'
>>> s[:]
'basic'
>>>

>>> s*3
'basicbasicbasic'

>>> len(s)
5
```

<u>Note:</u>

1) In Python the following data types are considered as Fundamental Data types

- int
- float
- complex
- bool
- str

2) In Python, we can represent char values also by using str type and explicitly char type is not available.

```
>>> c='a'
>>> type(c)
<class 'str'>
```

3) long Data Type is available in Python2 but not in Python3. In Python3 long values also we can represent by using int type only.

4) In Python we can present char Value also by using str Type and explicitly char Type is not available.