Chapter 4 : Data Handling

Data types

- Data types are means to identify the type of data and associated operations of handling it.
- Datatype can be many types eg character, integer, real, string etc.
- Built- in Core Data types
  - Numbers (int , float, complex)
  - String
  - List
  - Tuple
  - Dictionary

Data types

- Data types are means to identify the type of data and associated operations of handling it.

Different umbers data type are

- Integers
  - Integers (Signed)
  - Booleans
  -
- **Integers :-**
  - Integers are whole numbers such as 5,39,1917, 0 etc
  - They have no fractional parts.
  - Integers are represented in Python by numeric values with no decimal point.
  - Integers can be positive or negative e,g +12,-15,3000(missing + or-symbol means it is positive number).
  -

Type of integers

Integers (signed)---

- It is the normal integer representation of whole numbers.
- Integers in Python 3.x can be of any length, it is only limited by the memory available.
- Unlike other languages, Python 3.x provides single data type(int) to store any integer,

whether big or small.

- It is signed representation i.e the integers can be positive as well as negative.

Booleans

- These represent the truth values False and True.
- Boolean values False and True behave like the values 0 and 1 respectively.
- When we convert Boolean value False and True to a string the string '**False**' or '**True**' are
  Returned, respectively.
- The str( ) function converts a value to string.

```
>>> bool(0)
    False
>>>bool(1)
```

```
>>> str(False)
    False
>>>Str(True)
```

Floating Point Numbers

- A number having fractional part is a floating-point number. For example 3.14159 is floating point number.
- The number 14 is an integer but 14.0 is a floating point number.
- The fractional numbers can be written in two form :
  - **Fractional Form**(normal) eg 3500.75, 0.00005 etc
  - **Exponent Notation eg 3.50075E03, 0.5E-04, 1.479101E02 etc**

Eg **0.000000123**

1.23 E-7 the letter E is called exponent

- **Floating –point numbers have two advantages over integers** :
  - It represent values between the integers.
  - It represent a much greater range of values.
- **Floating –point numbers have one disadvantage over integers**
  - Floating-point operations are usually slower than integer operations

Complex number

- A complex number is a number of the form A+Bi where I is the imaginary number.
- A complex number is made up of both **real and imaginary components**.
- In complex number A+Bi A and B are real numbers and I is imaginary .
- If have a complex number z, where z= a+bi than **a** woulid be the real component and **b** would represent the imaginary component of z eg real component of z= 4 + 3i is **4** and the imaginary component would be **3**.

```
a=0+3.1 j

b=1.5+2j
```

```
>>>c=0+4.5j
>>>d=1.1+3.4j
>>>c
4.5 j
>>>d
```

```
>>>z=(1+2.56j)+(-4-3.56j)
>>>a
(-3-1j)
>>>z.real
-3.0
```

Strings

- A python strings is a sequence of characters and each character can be individually accessed using its index
- A string data type lets us to hold string data i.e any number of valid characters into a set of quotation marks.
- In Python 3.x each character stored in string is a Unicode character i.e all strings in Python 3.x are sequences of pure Unicode characters .
- Unicode is a system designed to represent every character from every language.
- A strings can hold any type of known characters i.e letters , numbers and special characters of any known scripted language.
- Eg of some legal strings

  “**abcd”, “1234”, ‘$%^&’,’????’**

	0	1	2	3	4	5	6	8
Subject	C	O	M	P	U	T	E	R	S

- 9 -8 -7 -6 -5 -4 -3 -2 -1

Thus,

Subject[0]='C' Subject[2]='m' Subject[6]='e' Subject[-1]='s' Subject[-7]='m' Subject[-9]='C'

Note :--The index (also called subscript sometimes) is the numbered position of a letter in the string. In Python, indices begin 0 onwards in the forward rirection and -1,-2 …in the backward direction.

List

- **A list in Python represents a list of comma-separated values of any datatype between square brackets e.g following are some lists.**
- **Eg [1,2,3,4,5]**
  ['a','e','i','o','u']
  ['Neha',102,79.5]
- **>>>a=[1,2,3,4,5]**
  >>>a
  [1,2,3,4,5]

  To change first value in list
  >>>a[0]=10
  >>>a
  [10,2,3,4,5]

  To change first value in list
  >>>a[2]=30
  >>>a
  [10,2,30,4,5]

Dictionary

- **Dictionary data type is another feature in Python's hat.**
- **The dictionary is an unordered set of comma-separated Key:value pairs within { } , with the requirement that with in a dictionary, no two keys can be the same (i.e there are unique keys within a dictionary).**
- **Eg**

{'a':1, 'e':2, 'i':3 ,'o':4, 'u':5}

```
>>>vowels ={'a':1, 'e':2, 'i':3 ,'o':4, 'u':5}
>>>vowels['a']
1
>>>vowels['u']
5
```

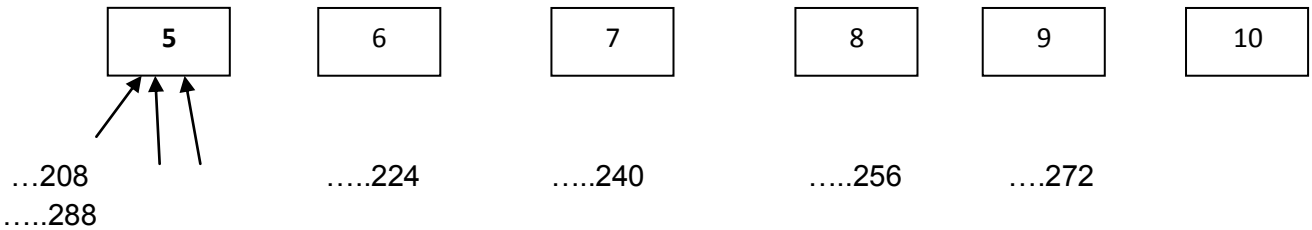
Note : Dictionaries shall be covered in details in later chapter

IMMUTABLE TYPES

- The immutable types are those that can never change their value in place.
  Eg integers , floating point numbers , Booleans , strings, tuples

- Immutability means that in the same memory address, new value cannot be stored as and when we want.

- Eg p=5
  q=p
  r=5
  .. # will give 5,5,5



p q r

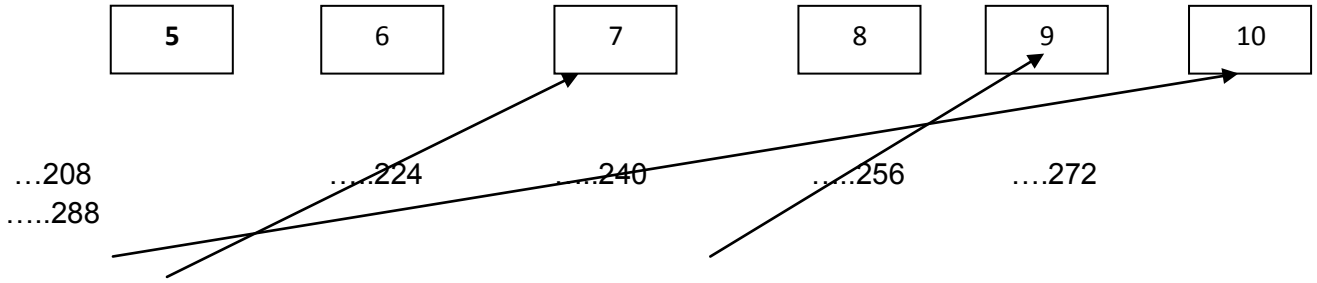
>>> id(5)
1457662208

>>> id(p)
1457662208

>>> id(q)
1457662208

>>> id(r)
1457662208

p=10
r=7
q=r



p q r

>>> id(10)
>>>id(r)
1457662288
1457662240

>>> id(p)
1457662288

>>> id(7)
1457662240

>>> id(q)
1457662240

>>>id(5)
1457662208

Now If we

>>>t=5
>>>id(t)
1457662208 here variables t has reference memory address same as initial reference memory address of variable p when it has value 5.

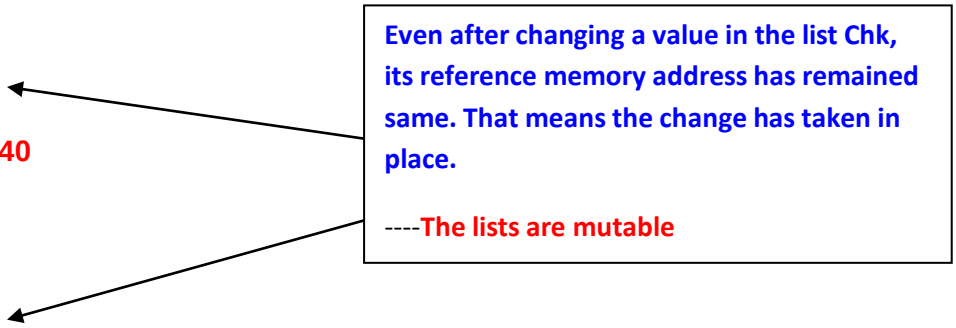
Note :---

- **In Python , variable-names are just the references to value-objects i.e data values.**
- **The variable-names are instead made to refer to new immutable integer object.**
- **The variable-names do not store values themselves i.e they are not storage containers.**

MUTABLE TYPES

- The mutable types are those whose values can be changed in place.
- Only three types are mutable in python .
  - Lists
  - Dictionaries
  - Sets (Not in syllabus)

```
>>>Chk=[2,4,6]
>>>id(Chk)
150195536
>>>Chk[1]=40
>>>Chk
[2, 40, 6]
id(chk)
150195536
```



Variable Internals

Objects :-

- Python is an object oriented language. Python calls every entity that stores any values or any type of data as an object.
- An object is an entity that has certain properties and that exhibit a certain type of behaviour eg integer values are objects – they hold whole numbers only and they have infinite precision (properties); they support all arithmetic operations(behaviour)
- All values are referred to as object in python. Similarly , we can say that a variable is also an object that refers to a value.

Every Python objects has three key attributes associated to it:

- The type of an object
  - The type of an object determines the operations that can be performed on the object. Built-in function type( ) returns the type of an object.
  - Eg >>> a=5
  - >>>type(5)

<class 'int'>

>>>type(a)

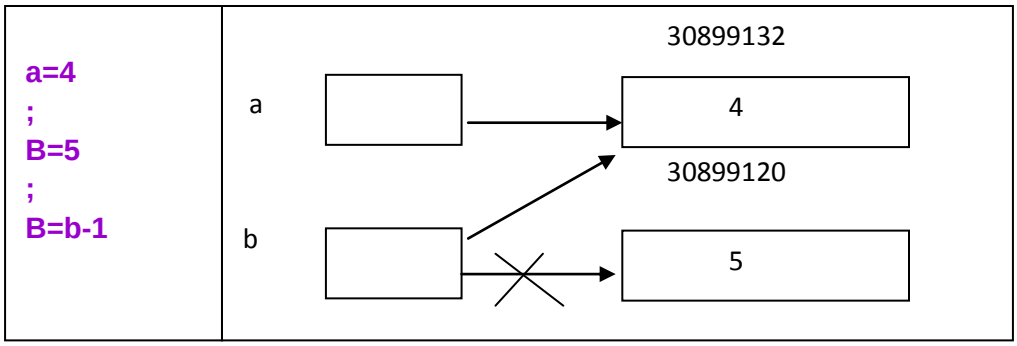
<class'int'>

- **The value of an object**
  - **It is the data –item contained in the object. For a literal , the value is the literal itself and for a variable the value is the data –item it(the variable) is currently referencing.**

```
>>>a=4

>>>print(4)

4

>>>print(a)

4
```

- **The id of an object**
  - **The id of an object is generally the memory location of the object.**
  - **Id returns the memory location of the object.**
  - **Built –in function id( ) returns the id of an object eg**

```
>>>id(4)
30899132
>>>a=4
>>>id(a)
30899132
```



Variable Internals

Objects :-

- **Python is an object oriented language. Python calls every entity that stores any values or any type of data as an object.**

- An object is an entity that has certain properties and that exhibit a certain type of behaviour eg integer values are objects – they hold whole numbers only and they have infinite precision (properties); they support all arithmetic operations(behaviour)
- All values are referred to as object in python. Similarly , we can say that a variable is also an object that refers to a value.

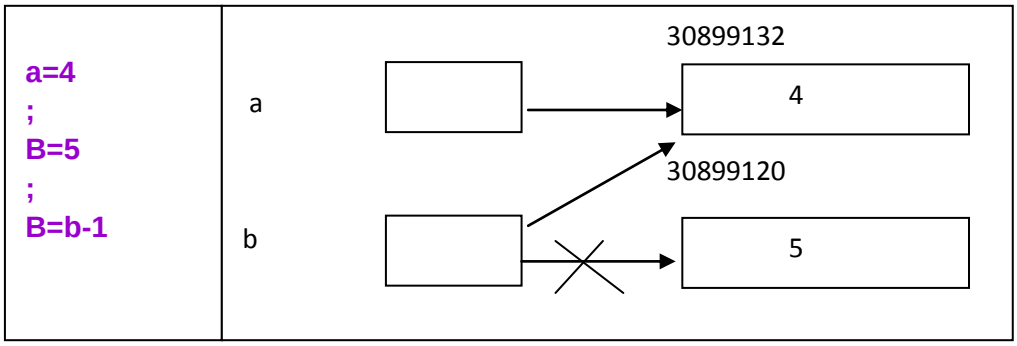
Every Python objects has three key attributes associated to it:

- The type of an object
  - The type of an object determines the operations that can be performed on the object. Built-in function type( ) returns the type of an object.
  - Eg >>> a=5
  - >>>type(5)
    <class 'int'>
    >>>type(a)
    <class'int'>
- The value of an object
- It is the data –item contained in the object. For a literal , the value is the literal itself
  - and for a variable the value is the data –item it(the variable) is currently referencing.

```
>>>a=4
>>>print(4)
4
>>>print(a)
4
```

- The id of an object
  - The id of an object is generally the memory location of the object.
  - Id returns the memory location of the object.
  - Built –in function id( ) returns the id of an object eg

```
>>>id(4)
30899132
>>>a=4
>>>id(a)
30899132
```



Operators

"The symbols that trigger the operation / action on data are called Operators, and the data on which operation is being carried out i.e the objects of operation(s) are referred to as operands.

Arithmetic Operators

Python use arithmetic operators for basic calculations like

- + addition
- -subtraction
- *multiplication
- / division
- // floor division
- % remainder
- ** exponentiation

Unary Operators

- The operators that act on one operand are referred to as Unary Operators.
- Unary + The operators unary '+'precedes an operand. The operand (the value on which the
  Operator operates) eg a=5 than +a means 5
- Unary- The operators unary '-'precedes an operand. The operand of the unary – operator must have arithmetic type and the result is the negation of its operand's value eg if a=5 then –a means -5

Binary Operators

" Operators that act upon two operands are referred to as Binary Operators.

- **Addition Operator + eg 4 +2= result in 24**
- **Subtraction operator- eg 14-3= evaluates to 11**

- **Multiplication ***
  The * operator multiplies the values of its operands.
  Eg 3* 4 evaluates to 12
  b*4(where b=6) evaluates to 24
- **Division operator / in this / operator divides its first operand by the second operand and always returns the result as a float value e.g.**
  4/ 2 evaluates to 2.0
  100/2 evaluates to 10.0
- **Floor Division operator //**
  - ✓ **Python also offers another division operator //, which performs the floor division .**
  - ✓ **The floor division is the division in which only the whole part of the result is given in the output and the fractional part is truncated.**
  - ✓ **a=15.9 , b=3**
    a/b evaluates to 5.3
    a//b will evaluate to 5.0
- **Modulus operator %**
  - ✓ **The % operator finds the modulus (i.e, remainder but pronounced as mo-du-lo) of its first operand relative to the second. That is it produces the remainder of dividing the first operand by the second operand.**
  - ✓ **Eg 19 % 6 evaluates to 1, since 6 goes into 19 three times with a remainder 1.**

Augmented Assignment Operators

- We know that in Python has an assignment operator =which assigns the value specified on RHS to the variable/object on the LHS of =.
- Python also offers augmented assignment arithmetic operators , which combine the impact of an arithmetic operator with an assignment operator,
- Eg **a=a+b**
  We may write
  a+=b

- This operators can be used anywhere that ordinary assignment is used.
- Augmented assignment doesn't violate mutability.

Relational Operators

- **In the term relational operator , relational refers to the relationships that values (or operands) can have with one another.**
- **The relational operators determine the relation among different operands.**
- **Python provides six (06)** relational operators **for comparing values(thus also calledcomparison operators).**
- **If the comparison is true, the relational expression results into the Boolean value True and to Boolean value False, if the comparison is false.**
- **< less than , <= less than or equal to , == equal to**
- **> greater than ,>= greater than or equal to , != not equal to**
- **a=3 , b=13**
  a<b will return True

Relational Operators with Arithmetic Operators

a+5>c-2

Identity Operators

- **There are two identity operators in Python is and is not.**
- **The identity operators are used to check if both the operands reference the same object memory i.e the identity operators compare the memory locations of two objects and return True or False accordingly.**

Operator	Usage	Description
is	a is b	returns True if both its operands are pointing to same object(i.e both referring to same memory location) , returns False otherwise.
is not	a is not b	Returns True if both its operands are pointing to different objects (i.e both referring to different memory location return False otherwise

- **For eg**

```
>>>a=235

>>>b=240

>>>c=235

>>>a is b

False
```

Equality (==) and Identity (is)---important Relation

- **When the is operator returns True for two variables, it implicitly means that the equality operator will also return True.**
- **That is , expression a is b as True means that a==b will also be True, always.**

```
>>> a,b=235,235

print(a,b)

>>>a is b

True
```

Exception

```
>>>s1='abc'

>>>s2=input("Enter a string:")

Enter a string :abc

>>>s1==s2

True

>>>s1 is s2
```

The reson behind this behaviour is that there are a few cases where Python creates two different objects that both store the same value.

Logical Relational Operators

- **Python provides three logical operators to combine existing expressions**
- **OR, AND, NOT**
- **OR –Operator evaluates to True if either of its (relational) operands evaluates to True;**
  False if both operands evaluate to false.
- **AND operators evaluates to True if both of its (relational)operands evaluate to True; False if either or both operands evaluate to false.**
- **NOT operator returns always a Boolean True or False.**

- **Expressions-** An expression in Python is any valid combination of Atoms and operators.
  Atom- An atom is something that has a value. Identifiers , literals, strings, list, tuples , sets, dictionaries.
  Expression is composed of one or more operations.

- **Arithmetic expression**- If an expression formed using arithmetic operators , it is an arithmetic expression. Arithmetic expressions can either be pure integer expression or pure real expression, sometimes a mixed expression can also be formed which is a mixture of both.
- **Relational (or logical)expression-** The expressions that result into false or true are called Boolean expressions. The Boolean expressions of constants, variables and logical and relational operators.
- **Type conversion-** The process of converting one predefined type into another is called conversion.
- **Implicit type conversion**- It is conversion performed by the compiler without programmer's intervention.
- Explicit type conversion Is user-defined that forces an expression to be of specific type. The explicit conversion of an operand to a specific type is called **Type Casting**.

Module:-

- **A python module is a file which contains some variables. And constants , some functions, objects etc. defined in it, which can be used in another Python programs by importing it.**
- **Other than built-in functions, Python makes available many more functions through modules in its standard library.**
- **Python standard library is a collection of many modules for different functionalities e.g module time offers time related functions; module string offers functions for string manipulations and so on.**

Working with math module

- **>>>import math**
  >>>math.pow(3,4)
  81.0

Working with random module

- This module provides random –number generators.
- A random number in simple words means – a number generated by chance i.e randomly.
- The most common random numbers generator functions in random module are :

random()	It returns random floating point number N in the range[0.0,1.0], i.e 0.0<=N<1.0. the number always be less than 1.0(only lower range-limit is inclusive.
Randint(a,b)	It returns a random integer N in the range (a,b)i.e , a<=N<=b (both range –limits are inclusive). It always generate the integer.
Randrange(,start>,<stop>,<step>)	It returns random numbers from range start…stop with step value.

>>>print(random.random())

>>>print(random.randint(15,35)) >>>print(random.randrange(11,45,4)

Using statistics Module

- The statistics module of the Python Standard Library provides many statistics functions such as mean( ), median( ) , mode( ) etc.

statistics.mean(<seq>)	It returns the average value of the set/sequence of values passed
statistics.median(<seq>)	It return the middle value of the set/sequence of values passed.
Statistics.mode	It returns the most often repeated value of the set/sequence of values passed.

```
>>> import statistics
>>>seq=[5,6,7,5,6,5,5,9,11,12,23,5]
>>>statistics.mean(seq)
8.25
>>>statistics.median(seq)
6.0
>>> statistics.median(seq)
```