ICTACADEMY

BASICS OF PYTHON

PRESENTED BY:
NITISH VIG

Agenda

Basics

Datatypes

Variables, keywords, identifiers

Python I/O

Import

Operations

BASICS

Basics

- The compiler converts the python program into another code called **Byte Code.**

- Byte Code represents a fixed set off instructions that represents all operations like Arithmetic, Comparison, Memory-related etc. which runs on any operating system and hardware.

- Any computer can execute only binary code which comprises 1’s / 0’s. To convert byte code to machine code, we use **Python Virtual Machine (PVM)**.

- An interpreter translates the program source code line by line. JIT (Just In Time) compiler improve speed or execution of compiler program.

Basics

ICTACADEMY

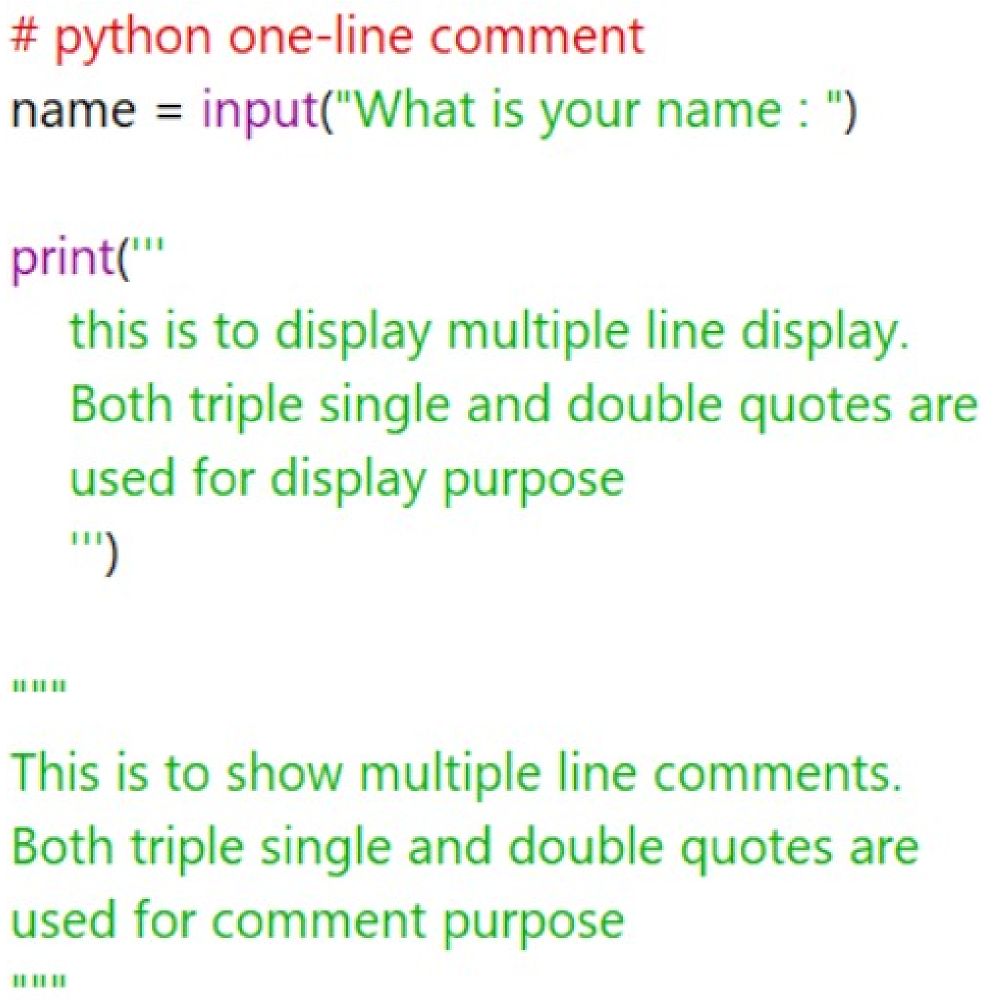
Single(‘’), Double(“”) quotes are used for single lines.

Tripe single(‘‘‘’’’), double (“““”””) quotes are used for multiple lines display and comments.

# (pound) sign is used for single line comment.

Basics

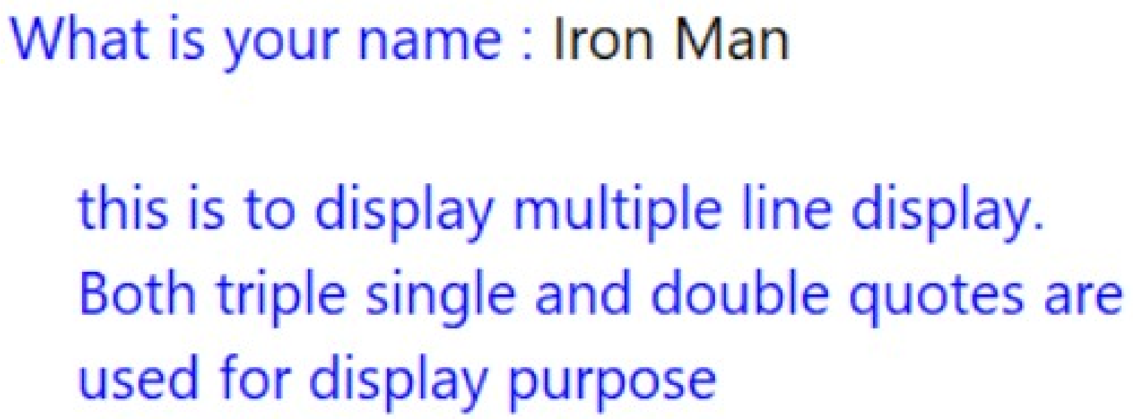
ICTACADEMY



```
# python one-line comment
name = input("What is your name : ")

print('''
    this is to display multiple line display.
    Both triple single and double quotes are
    used for display purpose
    ''')


"""
This is to show multiple line comments.
Both triple single and double quotes are
used for comment purpose
"""
```



```
What is your name : Iron Man

    this is to display multiple line display.
    Both triple single and double quotes are
    used for display purpose
```

ICTACADEMY

Basics

Escape Characters:

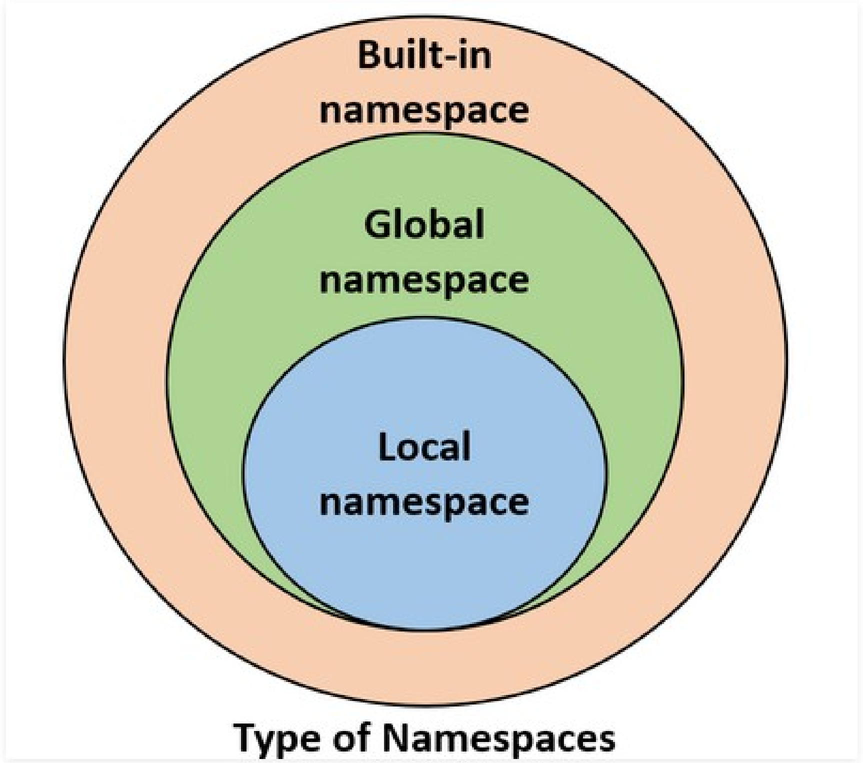
\	:	New line continuation
\\	:	Display a single line \
\’	:	Display multiple line quotes
\”	:	Display a single line quote
\b	:	Backspace
\r	:	Enter
\t	:	Horizontal Tab Space
\v	:	Vertical Tab Space
\n	:	New Line

Basics

ICTACADEMY

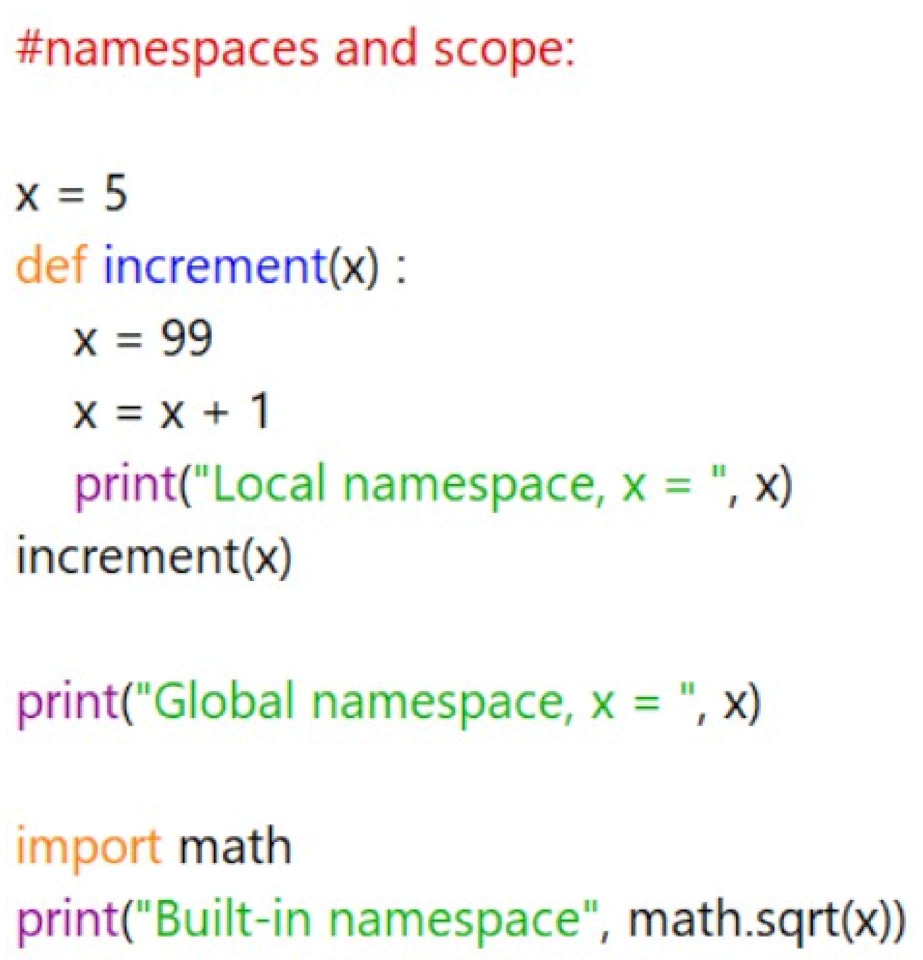
Namespaces:

- A namespace is a system to make sure that all the names in a program are unique.
- You might already know that everything in Python—like strings, lists, functions, etc.—is an object.
- Python implements namespaces as dictionaries.
- There is a name-to-object mapping, with the names as keys and the objects as values.



Basics

Namespaces:

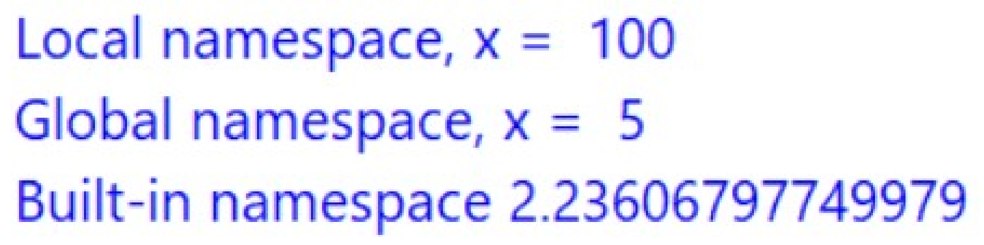


```python
#namespaces and scope:

x = 5
def increment(x) :
    x = 99
    x = x + 1
    print("Local namespace, x = ", x)
increment(x)

print("Global namespace, x = ", x)

import math
print("Built-in namespace", math.sqrt(x))
```



```
Local namespace, x =  100
Global namespace, x =  5
Built-in namespace 2.23606797749979
```

ICTACADEMY

KEYWORDS AND IDENTIFIERS

Python Keywords

- Keywords are the reserved words in Python.
- We cannot use a keyword as a variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language.
- In Python, keywords are case sensitive.

Python Keywords

ICTACADEMY

FALSE	class	finally	is	return
None	continue	for	lambda	try
TRUE	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Python Identifiers

An identifier is a name given to entities like class, functions, variables, etc. It helps to differentiate one entity from another.

Rules for writing identifiers:

1. Identifier can be of any length.

   Eg: pythonFundamentalAndDataScience ; DataPython; x

2. Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore _.

   Eg: Names like myClass, var_1 and print_this_to_screen, all are valid example.

3. An identifier cannot start with a digit.

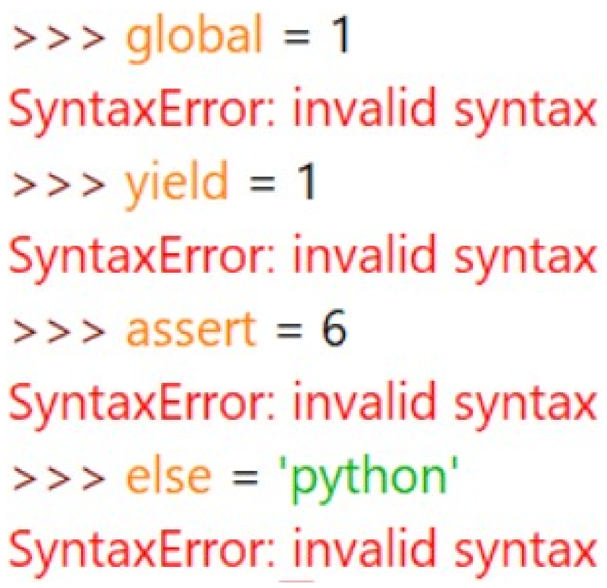
   Eg :1variable is invalid, but variable1 is perfectly fine.

Python Identifiers

Rules for writing identifiers

4. Keywords cannot be used as identifiers.

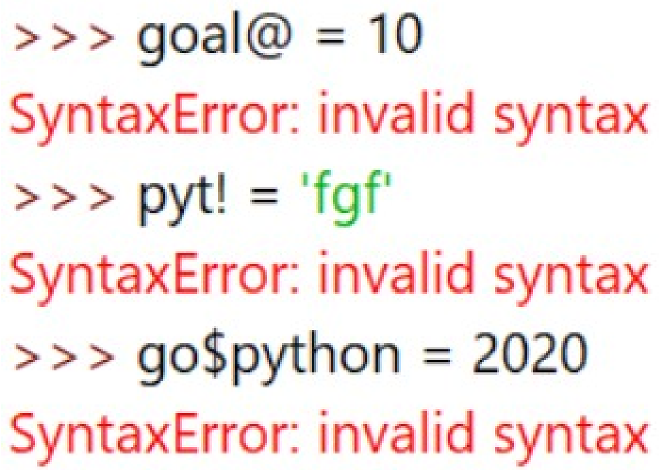
Eg:



```
>>> global = 1
SyntaxError: invalid syntax
>>> yield = 1
SyntaxError: invalid syntax
>>> assert = 6
SyntaxError: invalid syntax
>>> else = 'python'
SyntaxError: invalid syntax
```

5. We cannot use special symbols like **!**, **@**, **#**, **$**, **%** etc. in our identifier.

Eg:



```
>>> goal@ = 10
SyntaxError: invalid syntax
>>> pyt! = 'fgf'
SyntaxError: invalid syntax
>>> go$python = 2020
SyntaxError: invalid syntax
```

ICTACADEMY

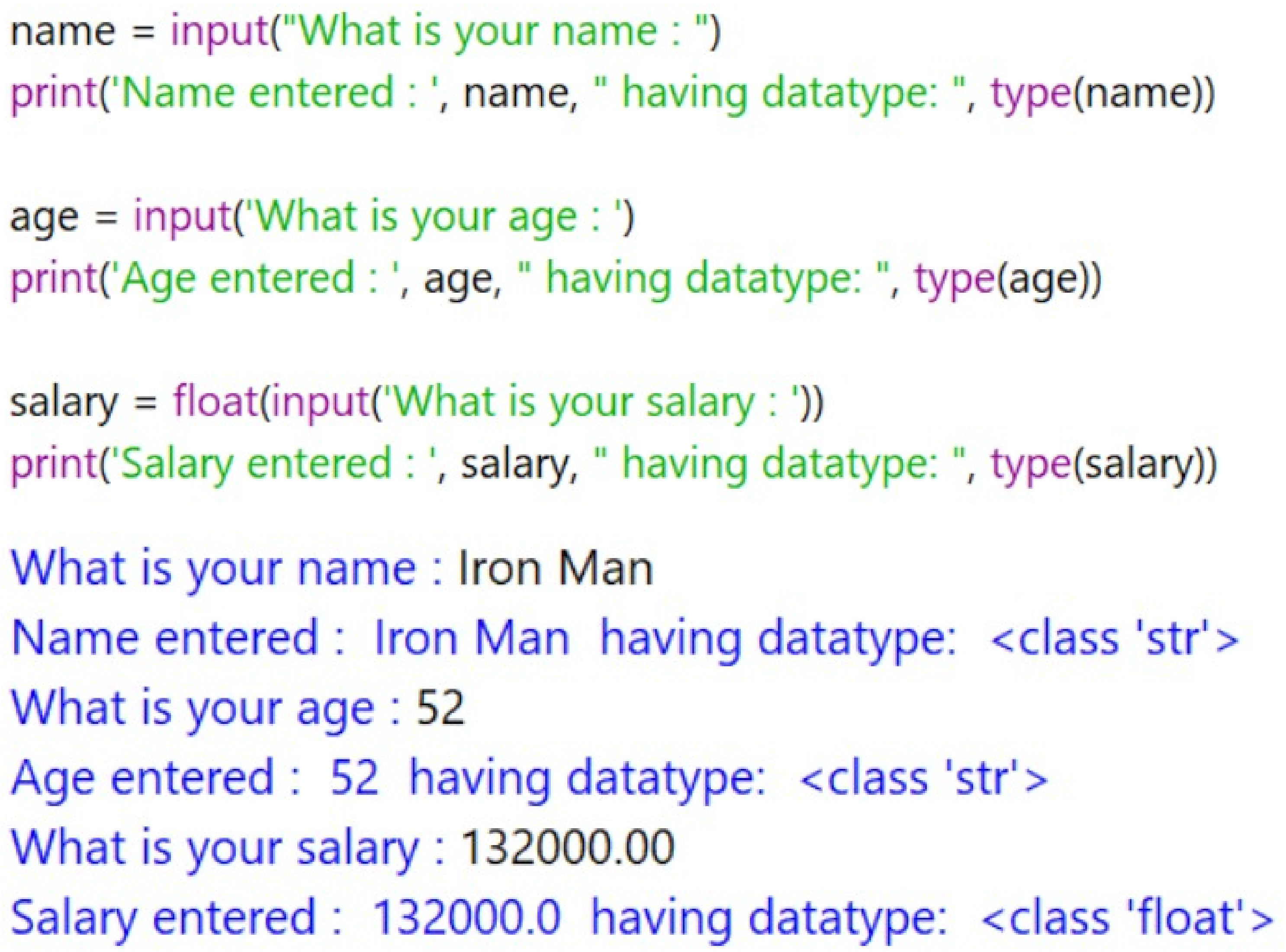
PYTHON
INPUT & OUTPUT

ICTACADEMY

Python input and output

- Keywords **input** is used to take input from the user. It can be any datatype.

- Keywords **print** is used to display output to the user. It can display text message or variable



```
name = input("What is your name : ")
print('Name entered : ', name, " having datatype: ", type(name))

age = input('What is your age : ')
print('Age entered : ', age, " having datatype: ", type(age))

salary = float(input('What is your salary : '))
print('Salary entered : ', salary, " having datatype: ", type(salary))
```

```
What is your name : Iron Man
Name entered :  Iron Man  having datatype:  <class 'str'>
What is your age : 52
Age entered :  52  having datatype:  <class 'str'>
What is your salary : 132000.00
Salary entered :  132000.0  having datatype:  <class 'float'>
```

ICTACADEMY

DATATYPES

Datatypes

- It represents the type of data stored in a variable / memory.
- There are built-in and user-defined datatype.

Built-In datatype

1. None type:

It doesn’t contain any value.

Mainly it is used in function to pass on empty values.
Eg: def sum():

In Boolean expression, it represent “False”

Built-In datatype

2. Numeric:

Integer (2, 5, -6, 101)

Float (3.22, -6.11, 0.009)

Complex (3 – 2j, 2 + 0.5j)

Built-In datatype

3. Boolean

True (1)

False (0)

A blank string “ ” is also represented as False.
Eg: abc = “ ”

Built-In datatype

4. Sequences: It represents a group of elements / items.

(a) String (str)

Syntax : *stringName = ‘’*

Represented by a group of characters

It can be enclosed in single(‘’) or double(“”) quotes

Eg:

```
    Name = ‘Nitish’
    Gender = “Male”
    abc = “I didn’t receive the gift”
    xyz = ‘ Hello”s world’
```

Built-In datatype

4. Sequences: It represents a group of elements / items.

(b) List:

Represented by a group of elements with different datatype, that can be modified directly.

Syntax : *list_1 = []*

Eg:

```
    city = ['Delhi', "Mumbai", "Chennai", 'Kolkata']
    fare = [2.5, 6, 6.05, 9.55]
    sym = [2.55, 'Python', "datatype", 'f', 9]
    dfy = [2.55, 'Python', "datatype", 'f', 9, [2.5, 6, 6.05, 9.55]]
```

ICTACADEMY

Built-In datatype

4. Sequences: It represents a group of elements / items.

(c) Tuple:

Represented by a group of elements with different datatype, that can be modified indirectly.

Syntax : *tup_1 = ()*

Eg:

```
    city = ('Delhi', "Mumbai", "Chennai", 'Kolkata')
    fare = (2.5, 6, 6.05, 9.55)
    sym = (2.55, 'Python', "datatype", 'f', 9)
    dfy = (2.55, 'Python', "datatype", 'f', 9, (2.5, 6, 6.05, 9.55))
```

Built-In datatype

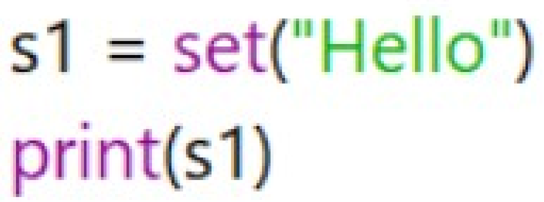
4. Sequences: It represents a group of elements / items.

(d) Sets:

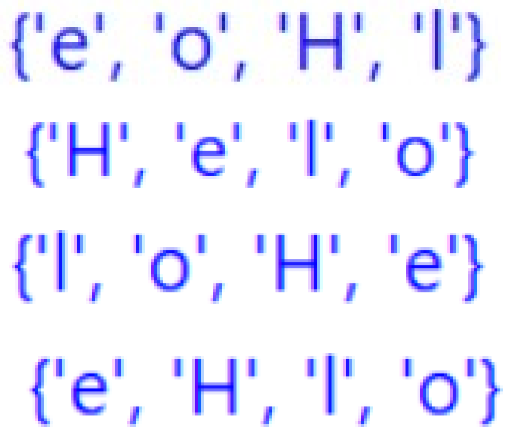
Represented by unordered collection of unique elements.
The elements may not appear in the same order they are written.
They can't be modified (eg, update, removal)

Syntax : *s1 = set(),or, {‘’}*

Eg:



```
s1 = set("Hello")
print(s1)
```



```
{'e', 'o', 'H', 'l'}
{'H', 'e', 'l', 'o'}
{'l', 'o', 'H', 'e'}
{'e', 'H', 'l', 'o'}
```

Built-In datatype

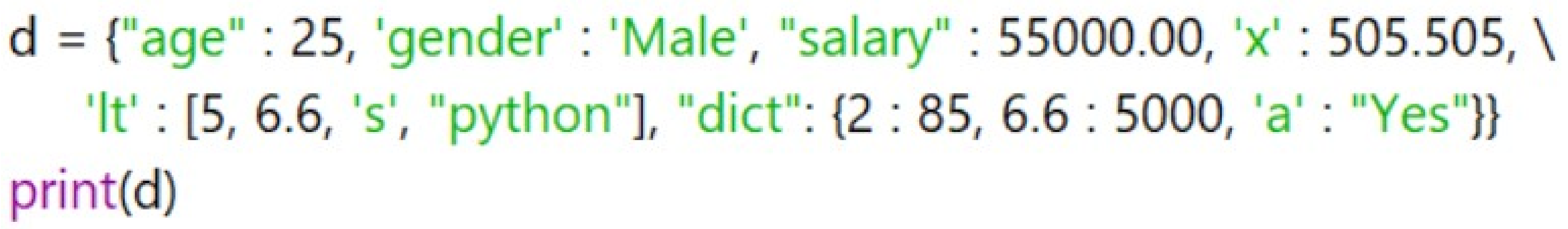
ICTACADEMY

5. Mappings:

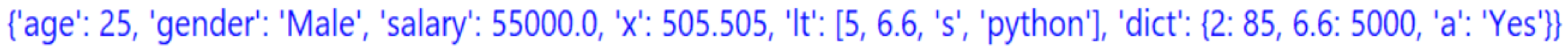
It represents a group values in key-value pair.
Keys are unique and they will be overwritten if updated twice.
Values can be duplicated. It can have strings, numeric, lists, dictionary

Syntax : dict = {}

Eg:

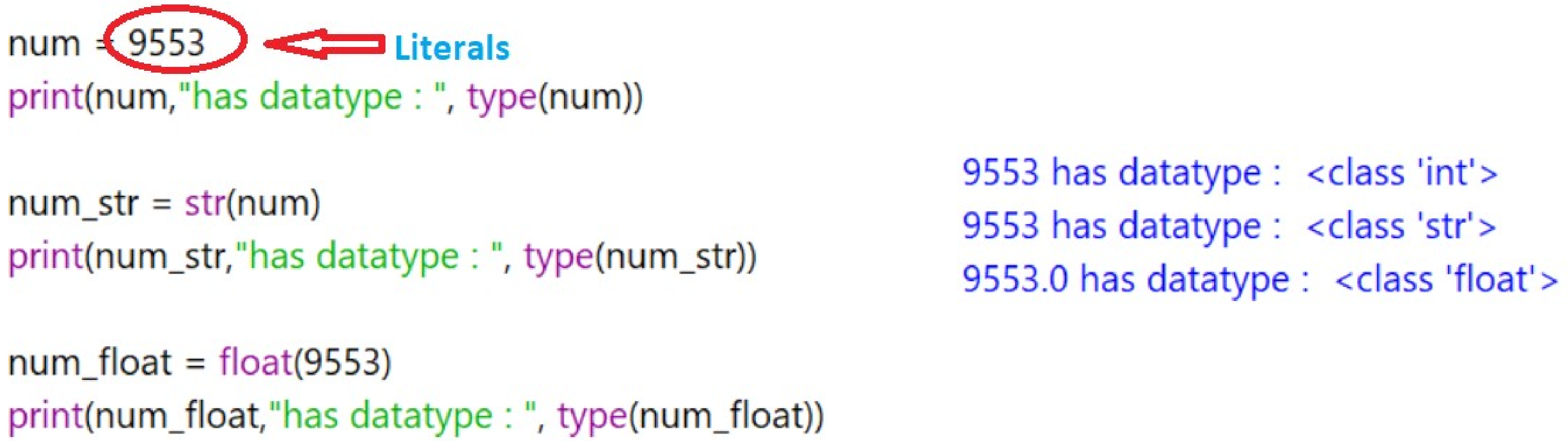


```
d = {"age" : 25, 'gender' : 'Male', "salary" : 55000.00, 'x' : 505.505, \
    'lt' : [5, 6.6, 's', "python"], "dict": {2 : 85, 6.6 : 5000, 'a' : "Yes"}}
print(d)
```



```
{'age': 25, 'gender': 'Male', 'salary': 55000.0, 'x': 505.505, 'lt': [5, 6.6, 's', 'python'], 'dict': {2: 85, 6.6: 5000, 'a': 'Yes'}}
```

User-defined datatype (Type – Casting)

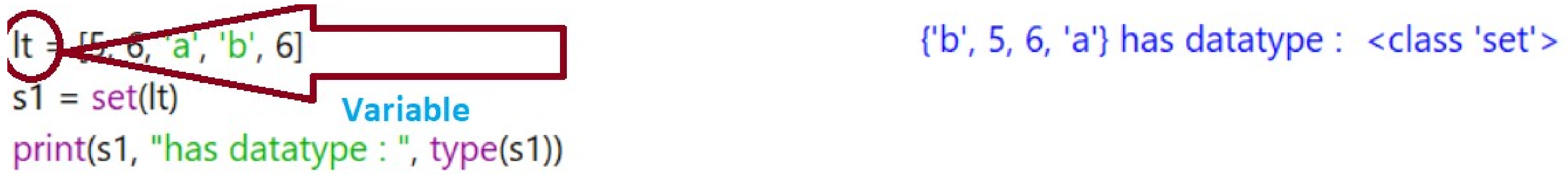


```
num = 9553
print(num,"has datatype : ", type(num))

num_str = str(num)
print(num_str,"has datatype : ", type(num_str))

num_float = float(9553)
print(num_float,"has datatype : ", type(num_float))
```

```
9553 has datatype :  <class 'int'>
9553 has datatype :  <class 'str'>
9553.0 has datatype :  <class 'float'>
```



```
lt = [5, 6, 'a', 'b', 6]
s1 = set(lt)
print(s1, "has datatype : ", type(s1))
```

```
{'b', 5, 6, 'a'} has datatype :  <class 'set'>
```

A Literal is a constant value that is stored into a variable in a program.

OPERATIONS

Operations

ICTACADEMY

1. Arithmetic
2. Assignment
3. Unary minus
4. Relational
5. Logical
6. Membership
7. Identity

Arithmetic

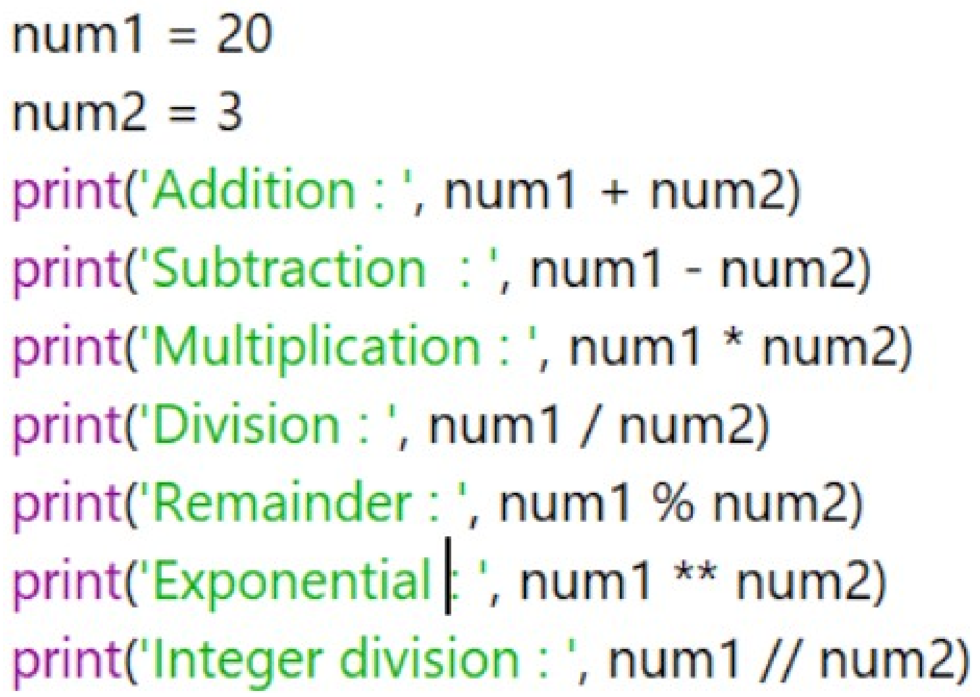
• Addition	(+)	+=
• Subtraction	(-)	-=
• Multiplication	(*)	*=
• Division	(/)	/=
• Remainder	(%)	%=
• Exponential	(**)	**=
• Integer / Floor division	(//)	//=

Order of operation :

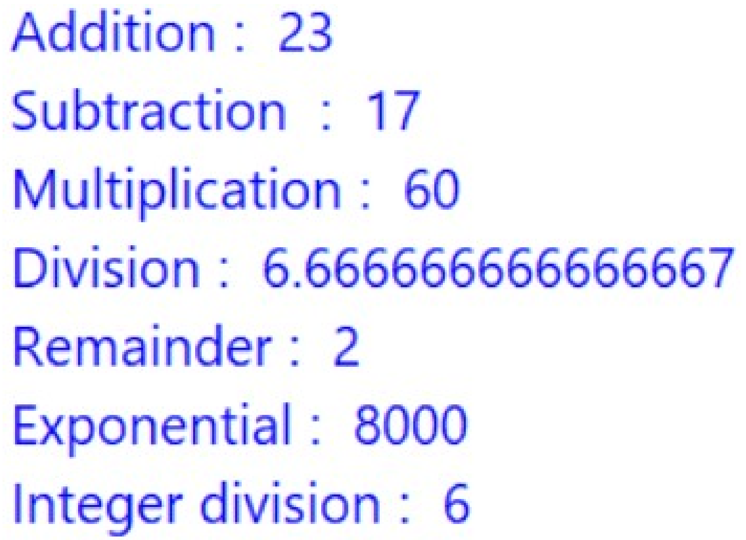
Exponential ▭ Multiplication ▭ Division ▭ Modulus ▭ Integer Division ▭ Addition ▭ Subtraction ▭ Assignment

i = i + 1 $\longleftrightarrow$ i += 1

Arithmetic



```python
num1 = 20
num2 = 3
print('Addition : ', num1 + num2)
print('Subtraction  : ', num1 - num2)
print('Multiplication : ', num1 * num2)
print('Division : ', num1 / num2)
print('Remainder : ', num1 % num2)
print('Exponential : ', num1 ** num2)
print('Integer division : ', num1 // num2)
```



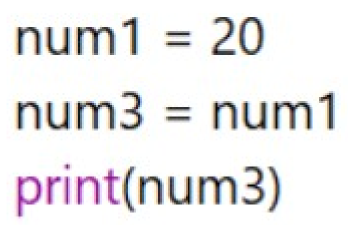
```
Addition :  23
Subtraction  :  17
Multiplication :  60
Division :  6.666666666666667
Remainder :  2
Exponential :  8000
Integer division :  6
```

Assignment

It stores the right side literal value to the left side variable.

The arithmetic operator before assignment operator means that after operation, value is stored in the left operand.

Represented by “ = ”



```
num1 = 20
num3 = num1
print(num3)
```

Output :
20

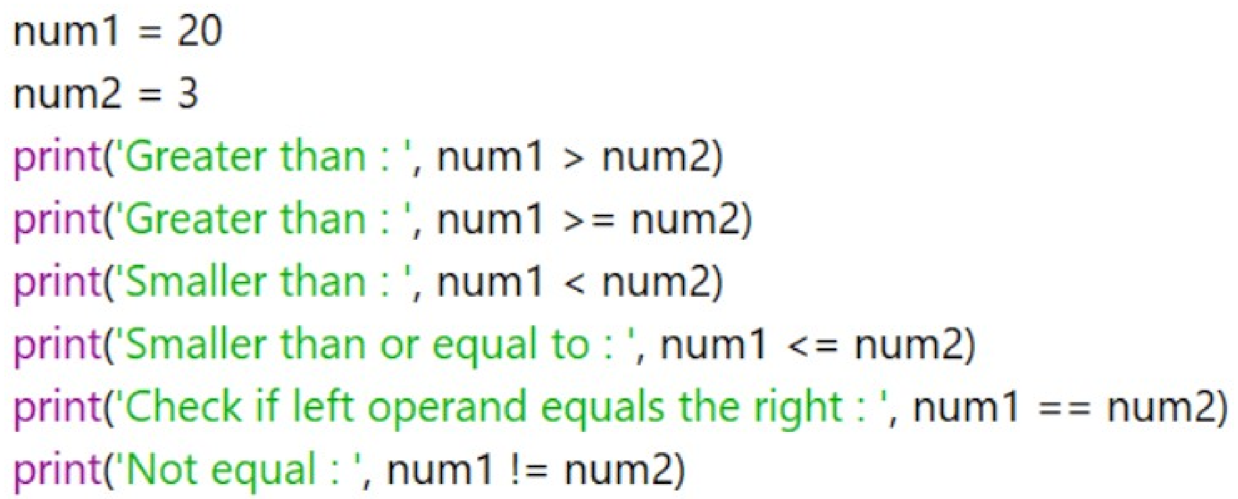
ICTACADEMY

Relational

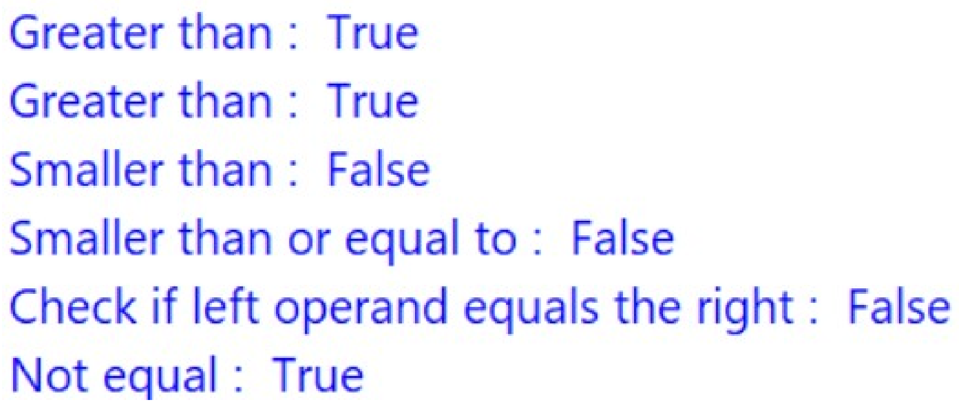
Sends True / False based on relation

• Greater than	(>)
• Greater than or equal to	(>=)
• Smaller than	(<)
• Smaller than or equal to	(<=)
• Check if left operand equals the right	(==)
• Not equal	(!=)
• is operator	(is)

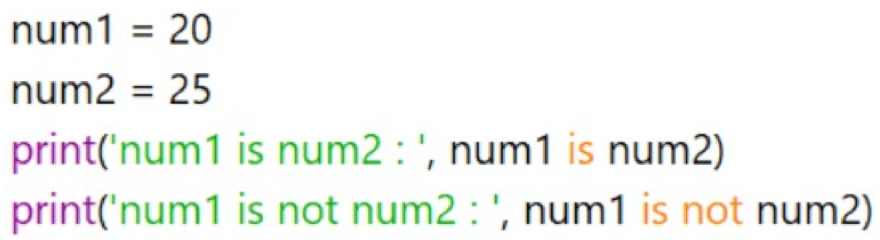
Relational



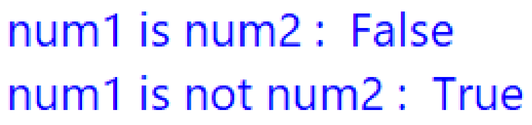
```python
num1 = 20
num2 = 3
print('Greater than : ', num1 > num2)
print('Greater than : ', num1 >= num2)
print('Smaller than : ', num1 < num2)
print('Smaller than or equal to : ', num1 <= num2)
print('Check if left operand equals the right : ', num1 == num2)
print('Not equal : ', num1 != num2)
```



```
Greater than :  True
Greater than :  True
Smaller than :  False
Smaller than or equal to :  False
Check if left operand equals the right :  False
Not equal :  True
```



```python
num1 = 20
num2 = 25
print('num1 is num2 : ', num1 is num2)
print('num1 is not num2 : ', num1 is not num2)
```



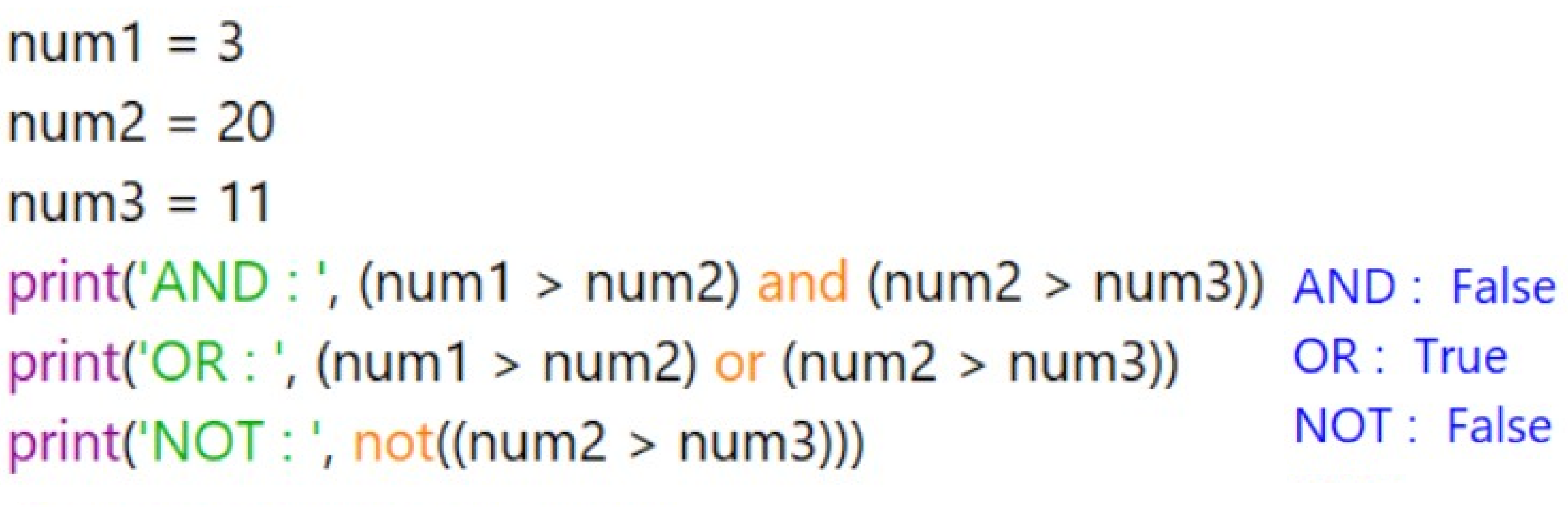
```
num1 is num2 :  False
num1 is not num2 :  True
```

(is, is not) is identity operator

Logical

Used to make a combination of one or more simple conditions.

Represented by: AND, OR, NOT



```
num1 = 3
num2 = 20
num3 = 11
print('AND : ', (num1 > num2) and (num2 > num3))
print('OR : ', (num1 > num2) or (num2 > num3))
print('NOT : ', not((num2 > num3)))
```

```
AND :  False
OR :  True
NOT :  False
```

ICTACADEMY

Membership Operator

This is used to check if element is present in the group.

Represented by “ in ” or “not in”

Eg,

```
list_1 = [10,20,30,40, “python”, “hello”]
element = 20
print(element in list_1)                    Output : True
```

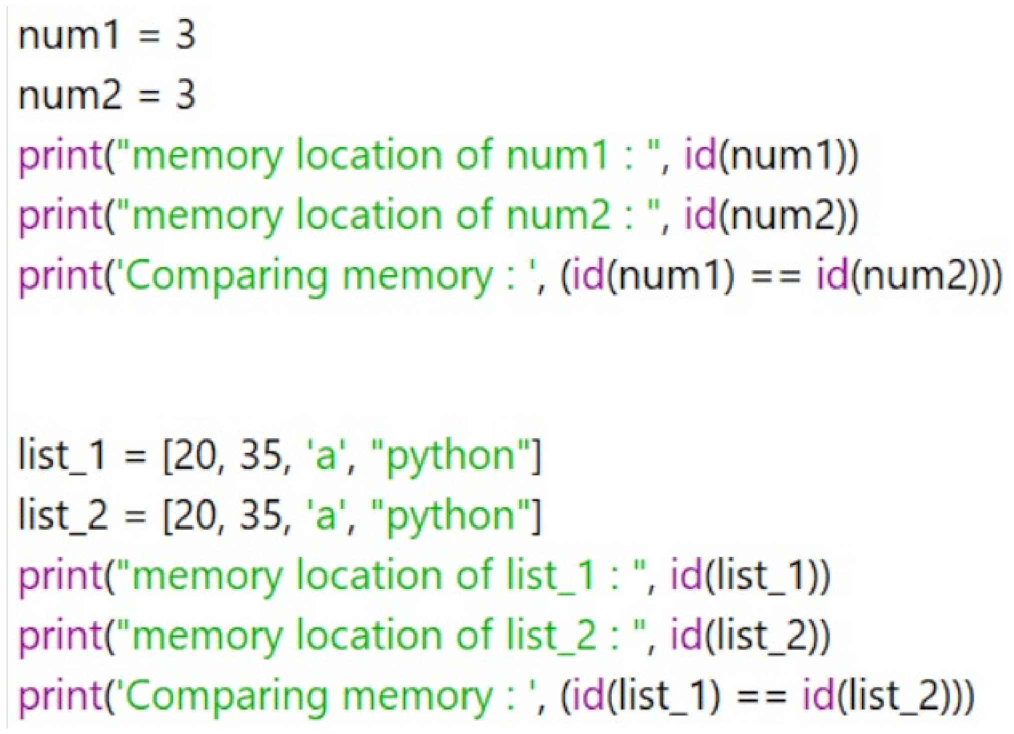
Eg,

```
list_1 = [10,20,30,40, “python”, “hello”]
element = “hell”
print(element in list_1)                    Output : False
```

Identity Operator

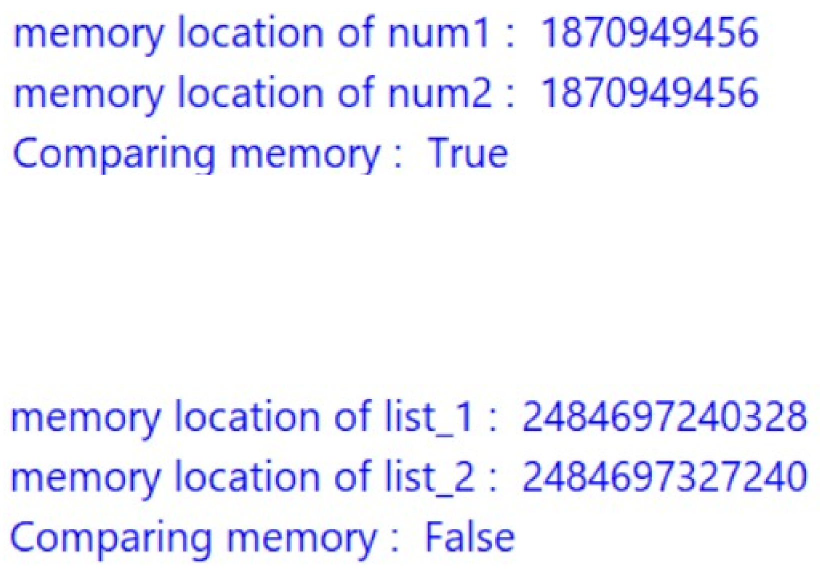
It represents the memory location of the object.
It can be used to compare memory locations of 2 objects

Represented by “ id() ”



```
num1 = 3
num2 = 3
print("memory location of num1 : ", id(num1))
print("memory location of num2 : ", id(num2))
print('Comparing memory : ', (id(num1) == id(num2)))


list_1 = [20, 35, 'a', "python"]
list_2 = [20, 35, 'a', "python"]
print("memory location of list_1 : ", id(list_1))
print("memory location of list_2 : ", id(list_2))
print('Comparing memory : ', (id(list_1) == id(list_2)))
```



```
memory location of num1 :  1870949456
memory location of num2 :  1870949456
Comparing memory :  True

memory location of list_1 :  2484697240328
memory location of list_2 :  2484697327240
Comparing memory :  False
```

ICTACADEMY

THANK YOU

Reach out to me at: **nitish@ictacademy.in**