ARYANZ.CO.IN

பைதான் கற்றுக்கொள்ளுங்கள் (தமிழில்)

Learn Python in Tamil

ARYANZ.CO.IN

பைதான் கற்றுக்கொள்ளுங்கள் (தமிழில்)

Python Type casting & Operators

in Tamil

Python Type cast

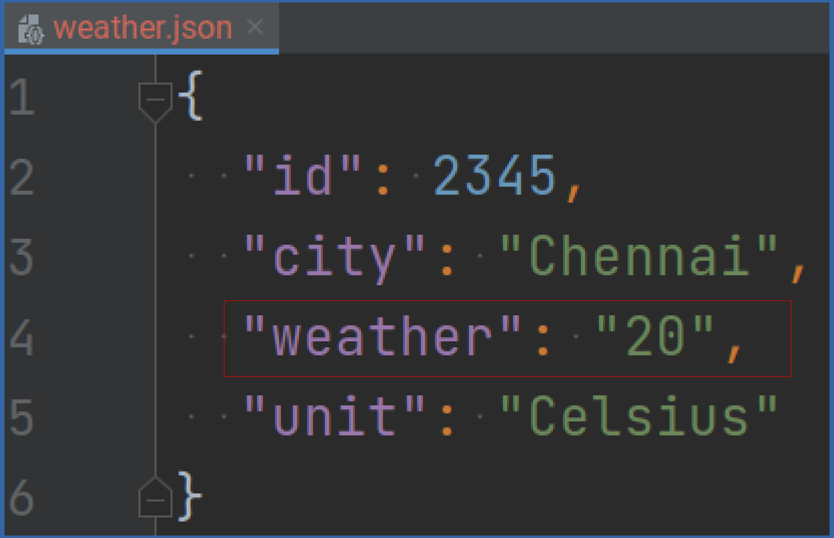
We know python is not strict with the data type.

Then why we need to type cast?

Consider you are using external weather api, whose response is in JSON (JavaScript Object Notation).

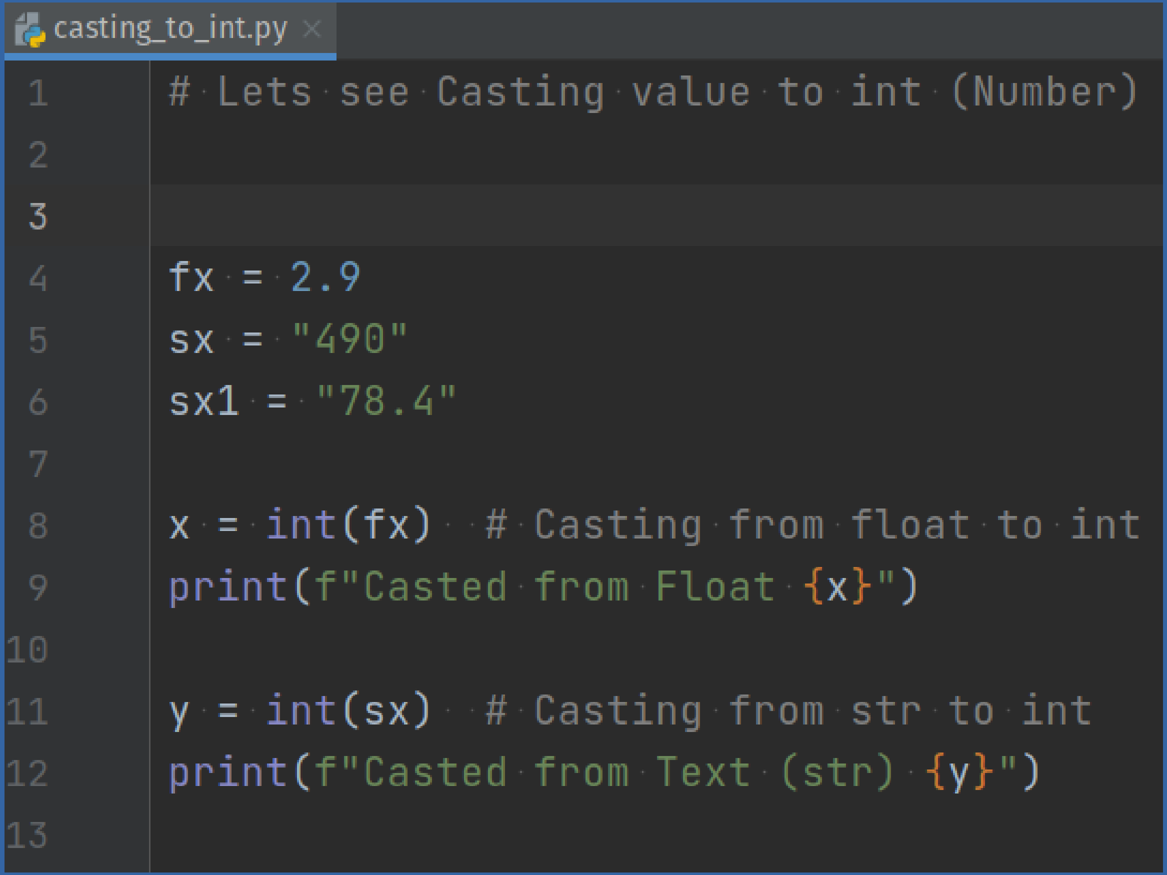
If you notice the JSON, the weather is in text/string. But in your program you need it to be int/number for converting from celsius to fahrenheit.

In such case we have to use type casting.



```
{
    "id": 2345,
    "city": "Chennai",
    "weather": "20",
    "unit": "Celsius"
}
```

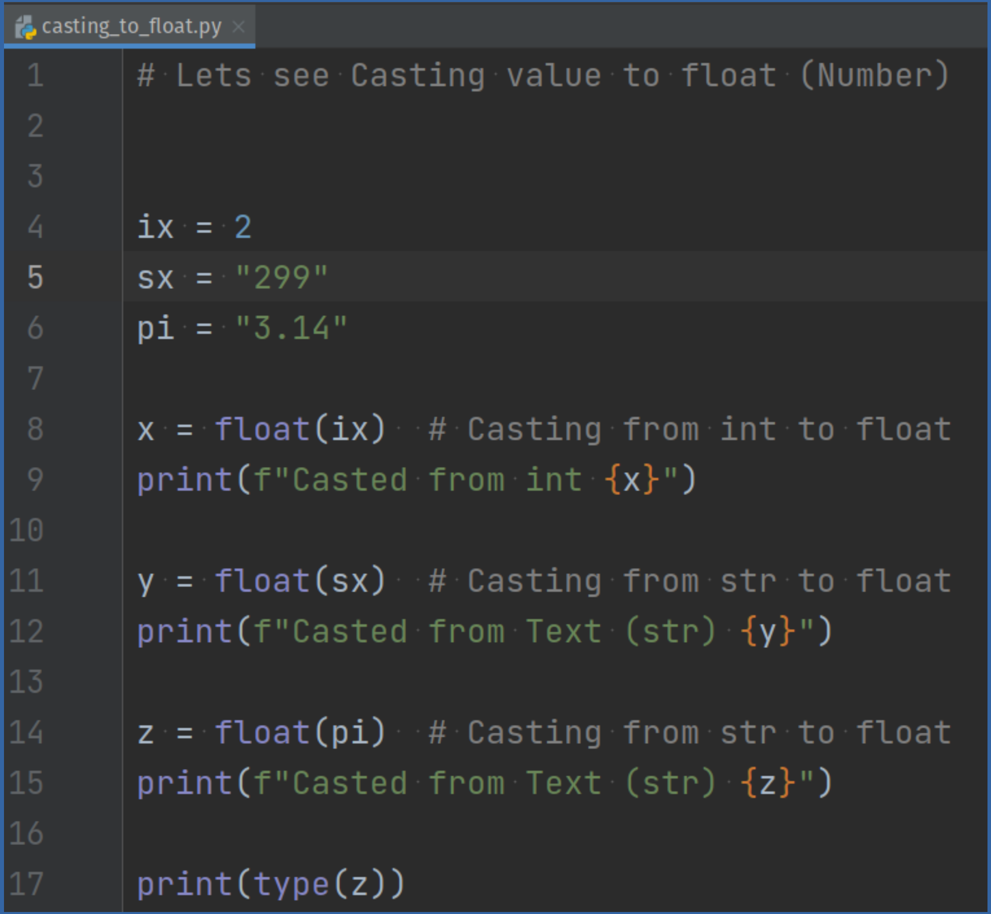
Python Type cast - int



casting_to_int.py

```python
# Lets see Casting value to int (Number)


fx = 2.9
sx = "490"
sx1 = "78.4"

x = int(fx)  # Casting from float to int
print(f"Casted from Float {x}")

y = int(sx)  # Casting from str to int
print(f"Casted from Text (str) {y}")

```

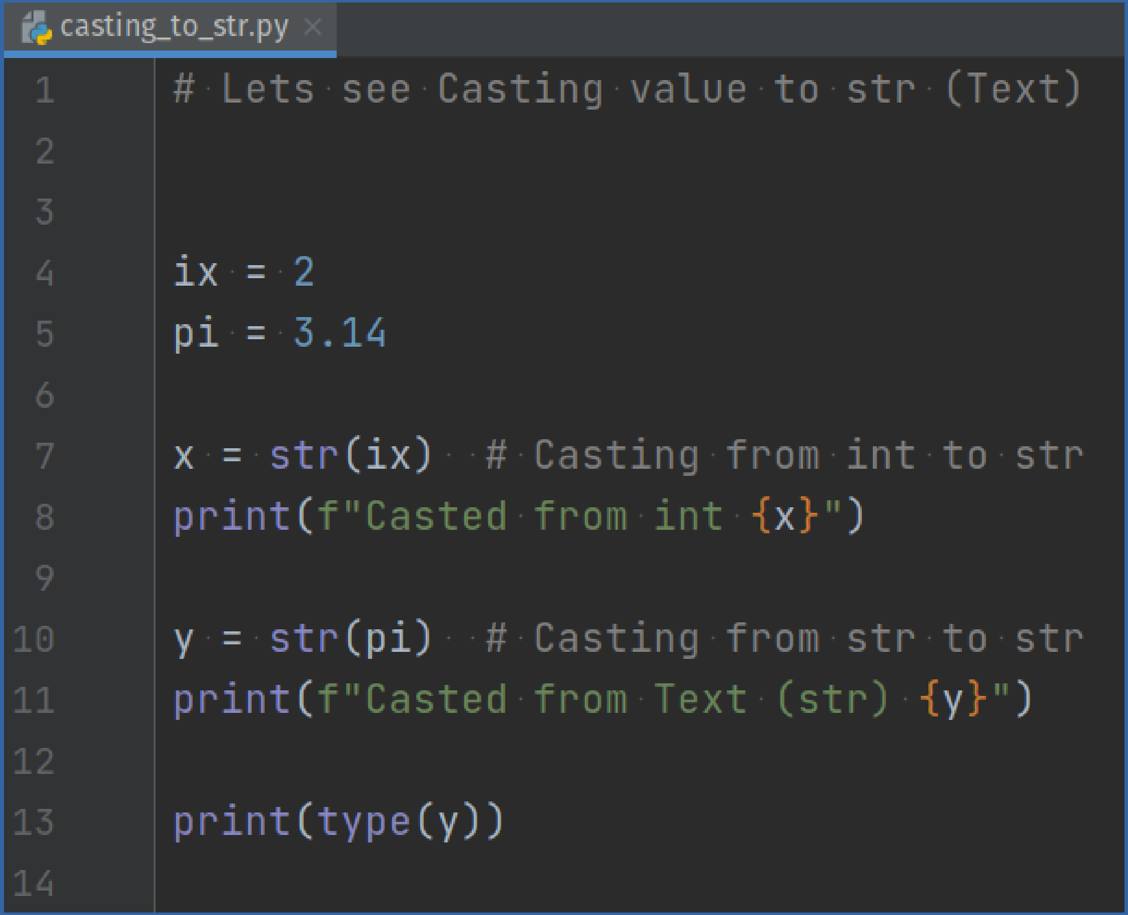
Python Type cast - Float



casting_to_float.py

```python
# Lets see Casting value to float (Number)


ix = 2
sx = "299"
pi = "3.14"

x = float(ix)  # Casting from int to float
print(f"Casted from int {x}")

y = float(sx)  # Casting from str to float
print(f"Casted from Text (str) {y}")

z = float(pi)  # Casting from str to float
print(f"Casted from Text (str) {z}")

print(type(z))
```

Python Type cast – str (text)



casting_to_str.py

```python
# Lets see Casting value to str (Text)


ix = 2
pi = 3.14

x = str(ix)  # Casting from int to str
print(f"Casted from int {x}")

y = str(pi)  # Casting from str to str
print(f"Casted from Text (str) {y}")

print(type(y))

```

ARYANZ.CO.IN

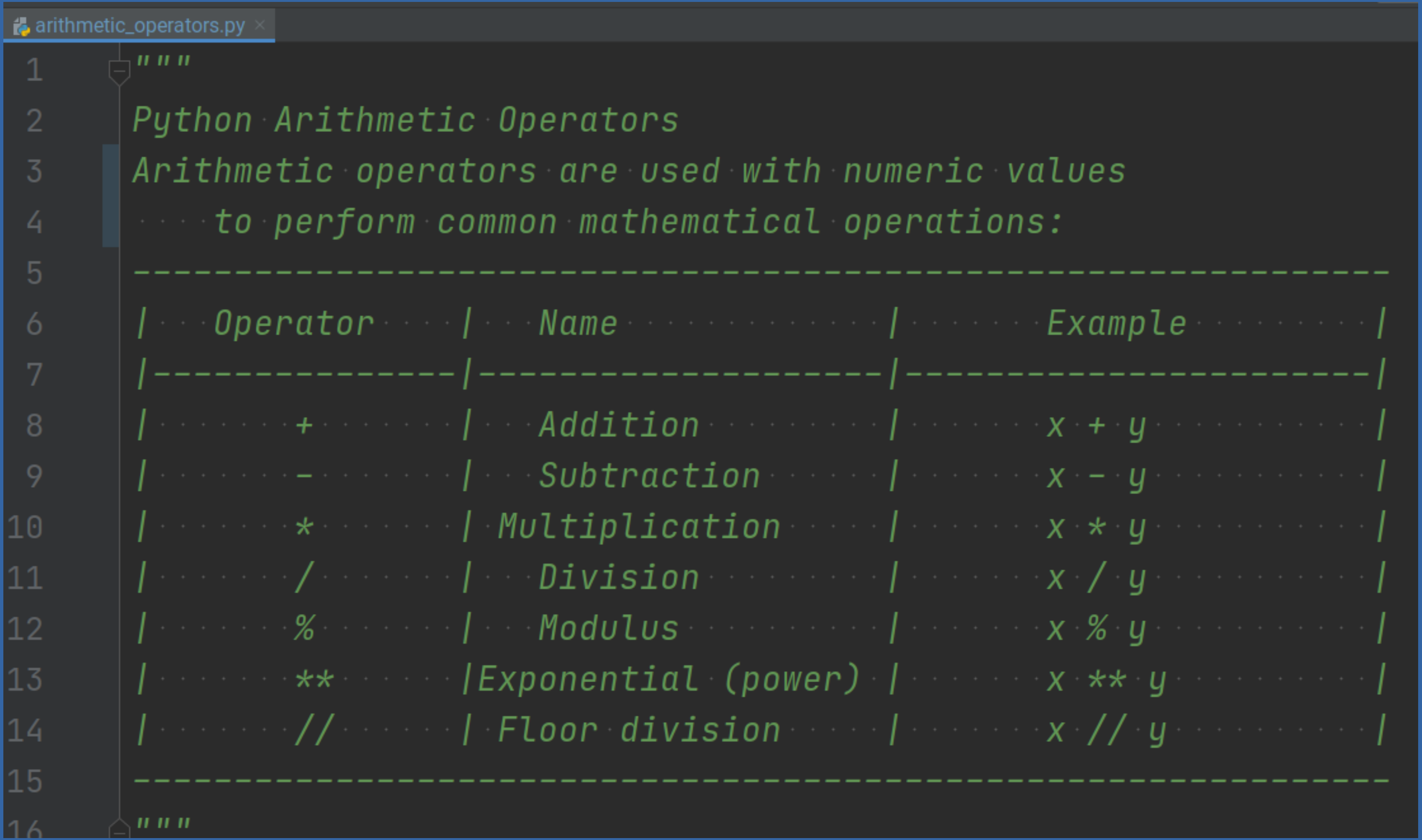
பைதான் கற்றுக்கொள்ளுங்கள் (தமிழில்)

Python operators

Python Operators

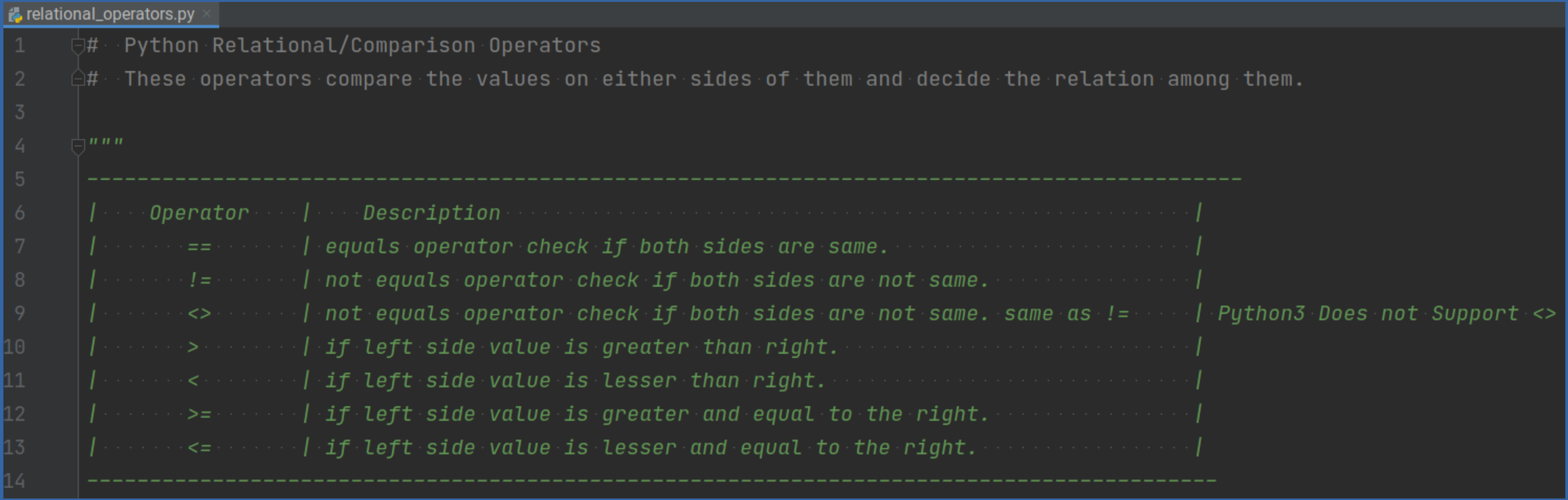
- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Python Arithmetic Operators



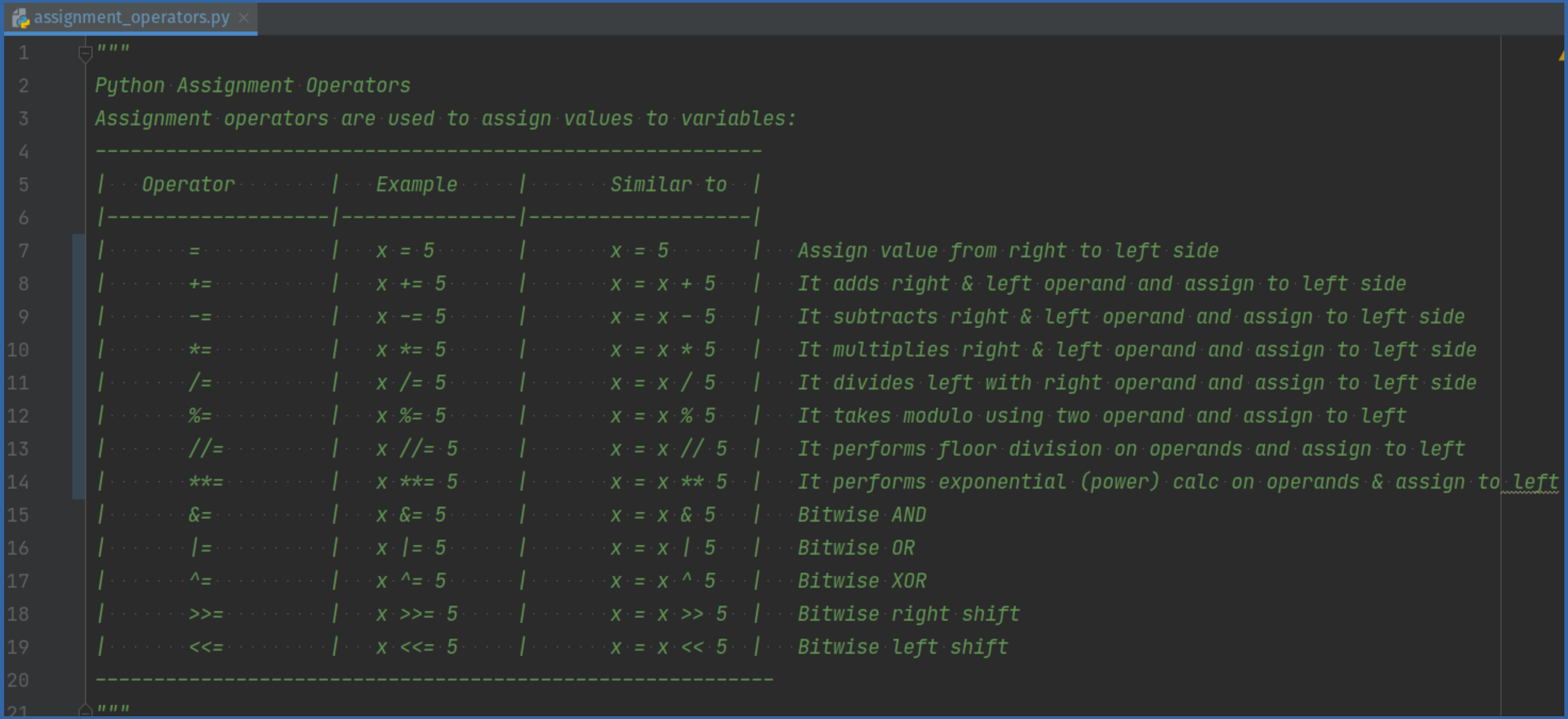
```
"""
Python Arithmetic Operators
Arithmetic operators are used with numeric values
    to perform common mathematical operations:
-------------------------------------------------------------
|   Operator    |   Name               |       Example       |
|---------------|----------------------|---------------------|
|       +       |   Addition           |       x + y         |
|       -       |   Subtraction        |       x - y         |
|       *       | Multiplication       |       x * y         |
|       /       |   Division           |       x / y         |
|       %       |   Modulus            |       x % y         |
|       **      |Exponential (power)   |       x ** y        |
|       //      | Floor division       |       x // y        |
-------------------------------------------------------------
"""
```

Python relational/comparison Operators


relational_operators.py

```
#  Python Relational/Comparison Operators
#  These operators compare the values on either sides of them and decide the relation among them.

"""
---------------------------------------------------------------------------------------------
|    Operator    |    Description                                                            |
|       ==       | equals operator check if both sides are same.                             |
|       !=       | not equals operator check if both sides are not same.                     |
|       <>       | not equals operator check if both sides are not same. same as !=          | Python3 Does not Support <>
|       >        | if left side value is greater than right.                                 |
|       <        | if left side value is lesser than right.                                  |
|       >=       | if left side value is greater and equal to the right.                     |
|       <=       | if left side value is lesser and equal to the right.                      |
-------------------------------------------------------------------------------------------
```

Python Assignment Operators



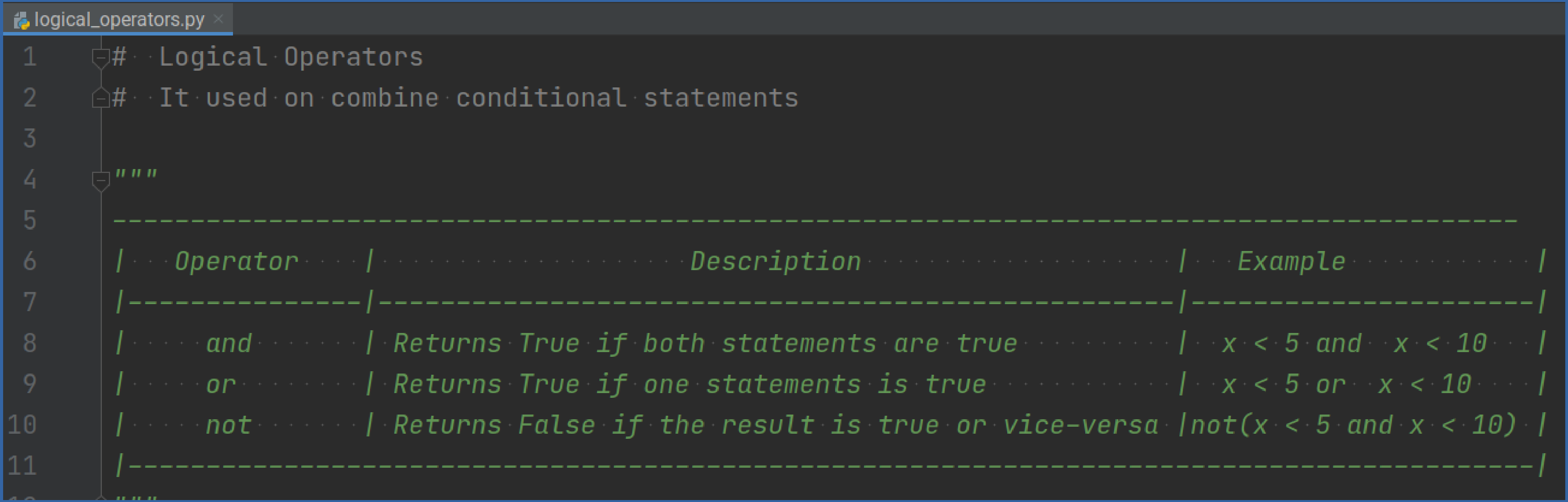
```
"""
Python Assignment Operators
Assignment operators are used to assign values to variables:
---------------------------------------------------------
|   Operator          |   Example      |       Similar to   |
|-------------------|---------------|-------------------|
|        =            |   x = 5        |       x = 5        |   Assign value from right to left side
|        +=           |   x += 5       |       x = x + 5    |   It adds right & left operand and assign to left side
|        -=           |   x -= 5       |       x = x - 5    |   It subtracts right & left operand and assign to left side
|        *=           |   x *= 5       |       x = x * 5    |   It multiplies right & left operand and assign to left side
|        /=           |   x /= 5       |       x = x / 5    |   It divides left with right operand and assign to left side
|        %=           |   x %= 5       |       x = x % 5    |   It takes modulo using two operand and assign to left
|        //=          |   x //= 5      |       x = x // 5   |   It performs floor division on operands and assign to left
|        **=          |   x **= 5      |       x = x ** 5   |   It performs exponential (power) calc on operands & assign to left
|        &=           |   x &= 5       |       x = x & 5    |   Bitwise AND
|        |=           |   x |= 5       |       x = x | 5    |   Bitwise OR
|        ^=           |   x ^= 5       |       x = x ^ 5    |   Bitwise XOR
|        >>=          |   x >>= 5      |       x = x >> 5   |   Bitwise right shift
|        <<=          |   x <<= 5      |       x = x << 5   |   Bitwise left shift
----------------------------------------------------------
"""
```

ARYANZ.CO.IN

பைதான் **assignment operator** | **Python assignment operator**

python

Python Logical Operators


logical_operators.py

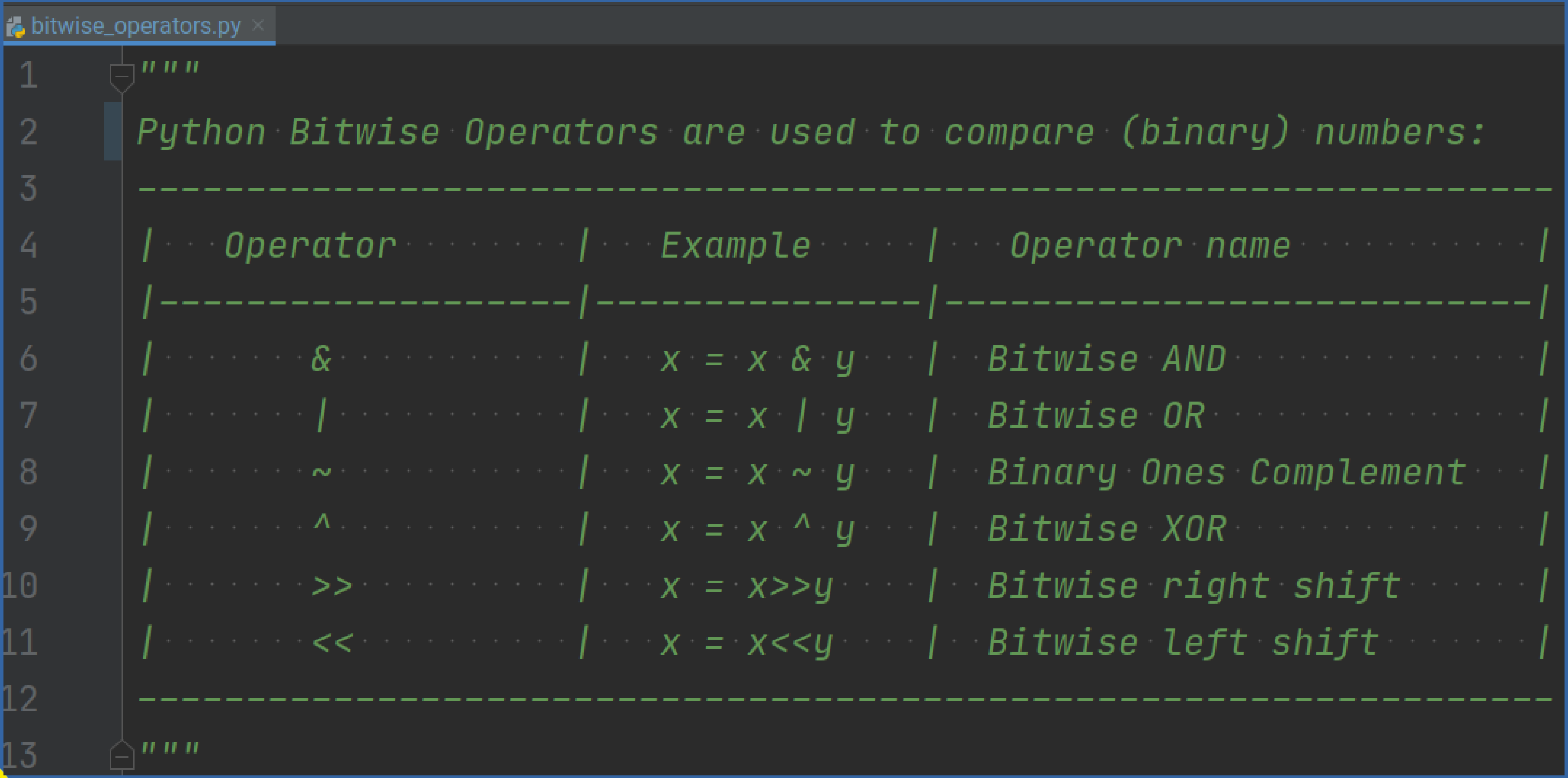
```
#  Logical Operators
#  It used on combine conditional statements

"""
------------------------------------------------------------------------------------------
|   Operator    |                    Description                    |   Example            |
|---------------|---------------------------------------------------|----------------------|
|     and       | Returns True if both statements are true          |  x < 5 and  x < 10   |
|     or        | Returns True if one statements is true            |  x < 5 or  x < 10    |
|     not       | Returns False if the result is true or vice-versa |not(x < 5 and x < 10) |
|------------------------------------------------------------------------------------------|
```

ARYANZ.CO.IN

பைதான் **Logical operator** | **Python logical operator**

python

Python Bitwise Operators



bitwise_operators.py

```
"""
Python Bitwise Operators are used to compare (binary) numbers:
------------------------------------------------------------------
|   Operator          |   Example      |   Operator name           |
|---------------------|----------------|---------------------------|
|       &             |   x = x & y    |  Bitwise AND              |
|       |             |   x = x | y    |  Bitwise OR               |
|       ~             |   x = x ~ y    |  Binary Ones Complement   |
|       ^             |   x = x ^ y    |  Bitwise XOR              |
|       >>            |   x = x>>y     |  Bitwise right shift      |
|       <<            |   x = x<<y     |  Bitwise left shift       |
------------------------------------------------------------------
"""
```

Python Bitwise AND - Truth table

AND Truth table

X	Y	X & Y
0	0	0
0	1	0
1	0	0
1	1	1

AND - Truth table

When both values are True then result will be True.

Lets see the table,

1. When Both the values are 0 (false) result will be 0(false).

2. When the value is 1(true) and another is 0(false), result will be 0(false).

3. When both the value are 1(true) then the result will be 1(true).

Python Bitwise OR - Truth table

OR Truth table

X	Y	X | Y
0	0	0
0	1	1
1	0	1
1	1	1

OR - Truth table

When any one of the values is True the result will also be True.

Lets see the table,

1. When Both the values are 0 (false) result will be 0(false).

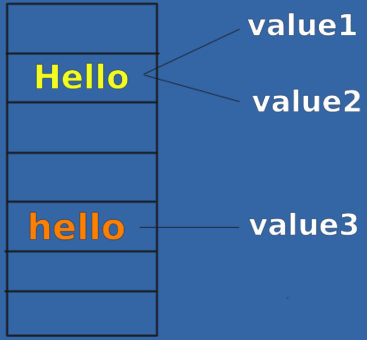
2. When the value is 1(true) and another is 0(false), result will be 1(true).

3. When both the value are 1(true) then the resule will be 1(true).

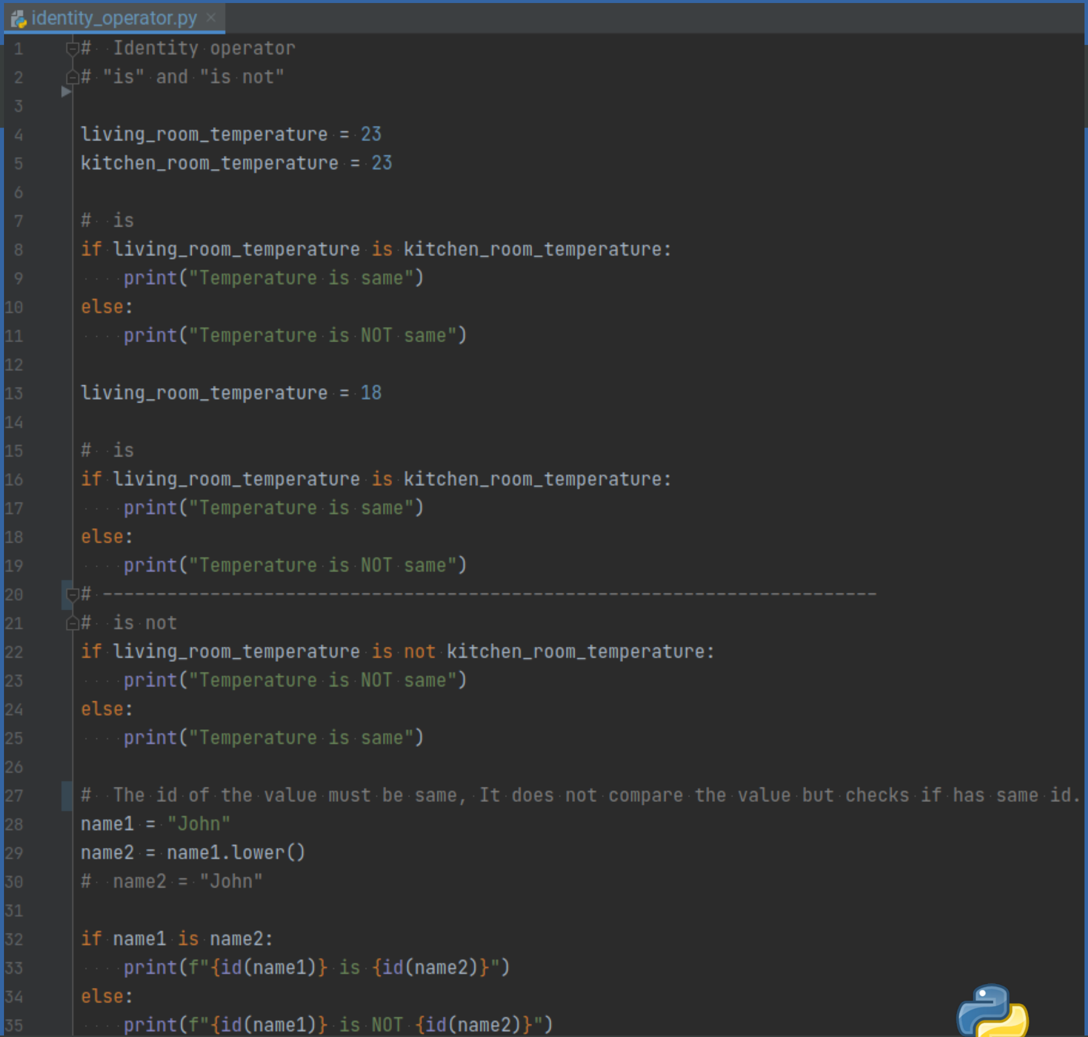
Python Identity Operators

Identity operators compare the memory locations of two objects.

There are two Identity operators explained below



Operator	Description
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.

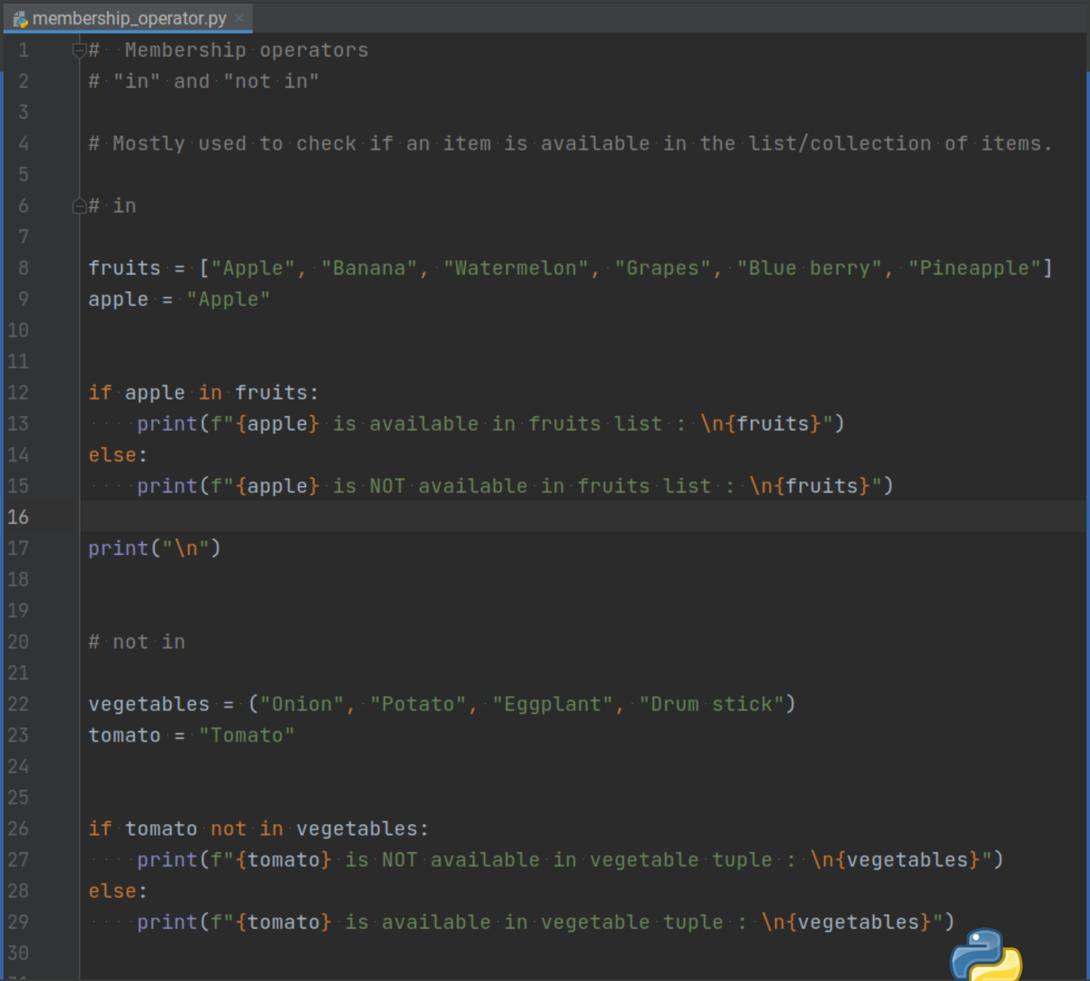

identity_operator.py

```python
# Identity operator
# "is" and "is not"

living_room_temperature = 23
kitchen_room_temperature = 23

# is
if living_room_temperature is kitchen_room_temperature:
    print("Temperature is same")
else:
    print("Temperature is NOT same")

living_room_temperature = 18

# is
if living_room_temperature is kitchen_room_temperature:
    print("Temperature is same")
else:
    print("Temperature is NOT same")
# ------------------------------------------------------------------------
# is not
if living_room_temperature is not kitchen_room_temperature:
    print("Temperature is NOT same")
else:
    print("Temperature is same")

# The id of the value must be same, It does not compare the value but checks if has same id.
name1 = "John"
name2 = name1.lower()
# name2 = "John"

if name1 is name2:
    print(f"{id(name1)} is {id(name2)}")
else:
    print(f"{id(name1)} is NOT {id(name2)}")
```

Python Membership Operators

Membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators, See below.

Operator	Description
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.



membership_operator.py

```python
# Membership operators
# "in" and "not in"

# Mostly used to check if an item is available in the list/collection of items.

# in

fruits = ["Apple", "Banana", "Watermelon", "Grapes", "Blue berry", "Pineapple"]
apple = "Apple"


if apple in fruits:
    print(f"{apple} is available in fruits list : \n{fruits}")
else:
    print(f"{apple} is NOT available in fruits list : \n{fruits}")

print("\n")


# not in

vegetables = ("Onion", "Potato", "Eggplant", "Drum stick")
tomato = "Tomato"


if tomato not in vegetables:
    print(f"{tomato} is NOT available in vegetable tuple : \n{vegetables}")
else:
    print(f"{tomato} is available in vegetable tuple : \n{vegetables}")

```

Python Type casting & Operators demo codes

You can checkout the Demo Codes from the below link

https://git.io/JtnlX

You can watch the Python course from the below link

https://youtu.be/cit6jKwKY1o

You can read our blog post here:

http://l.aryanz.co.in/ncbft3rz

ARYANZ.CO.IN

நன்றி

அடுத்த வீடியோவில் உங்களை சந்திப்போம்

See you in next video