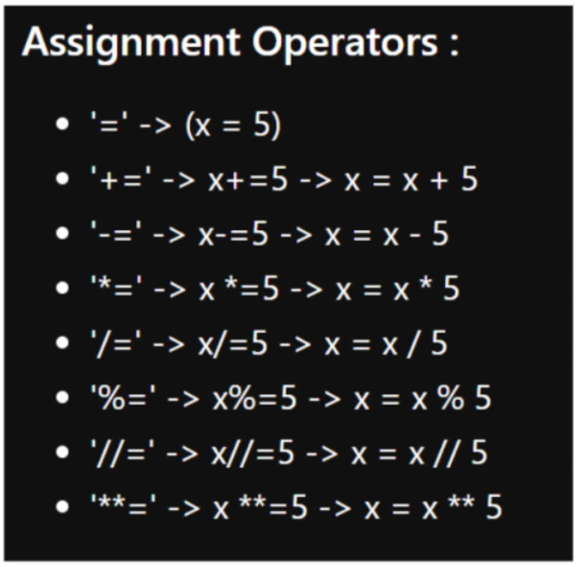
Operators & Strings

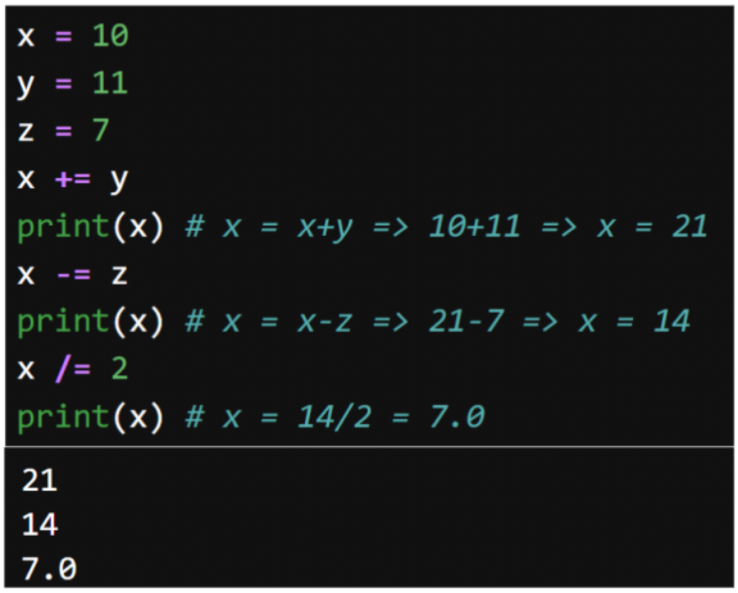
🎯 Session Objectives

- 🔧 Understand what operators are and why they are used
- 🧮 Explore different types of operators in Python
- 🔢 Learn about operator precedence and order of execution
- ⚠️ Understand constraints in programming
- 🔤 Understand string indexing and slicing
- 🔧 Explore common string methods and operations



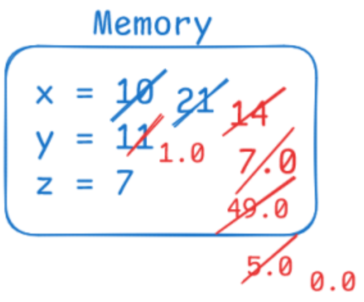
Assignment Operators :

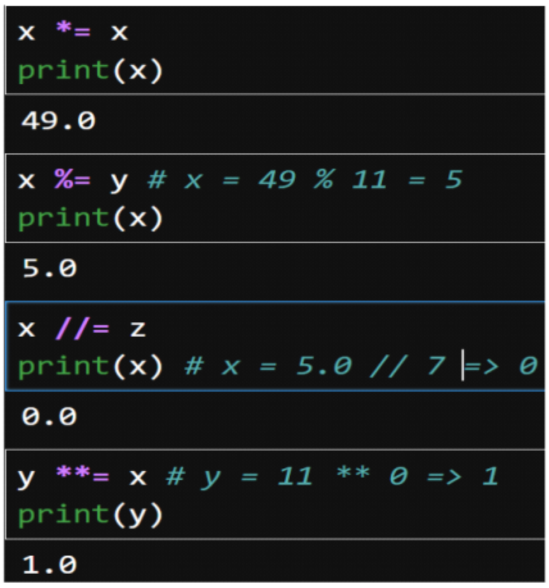
- '=' -> (x = 5)
- '+=' -> x+=5 -> x = x + 5
- '-=' -> x-=5 -> x = x - 5
- '*=' -> x *=5 -> x = x * 5
- '/=' -> x/=5 -> x = x / 5
- '%=' -> x%=5 -> x = x % 5
- '//=' -> x//=5 -> x = x // 5
- '**=' -> x **=5 -> x = x ** 5



```
x = 10
y = 11
z = 7
x += y
print(x) # x = x+y => 10+11 => x = 21
x -= z
print(x) # x = x-z => 21-7 => x = 14
x /= 2
print(x) # x = 14/2 = 7.0
```

```
21
14
7.0
```





```python
x *= x
print(x)
```

```
49.0
```

```python
x %= y # x = 49 % 11 = 5
print(x)
```

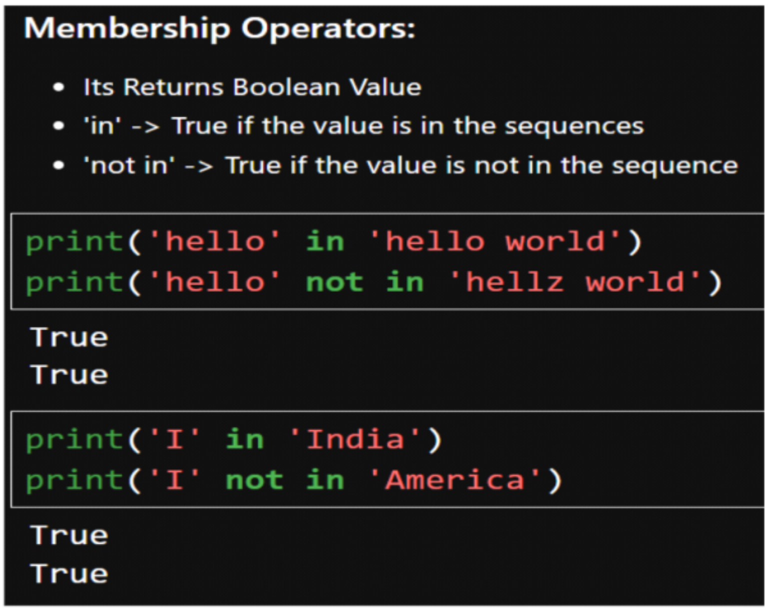
```
5.0
```

```python
x //= z
print(x) # x = 5.0 // 7 => 0
```

```
0.0
```

```python
y **= x # y = 11 ** 0 => 1
print(y)
```

```
1.0
```



Membership Operators:

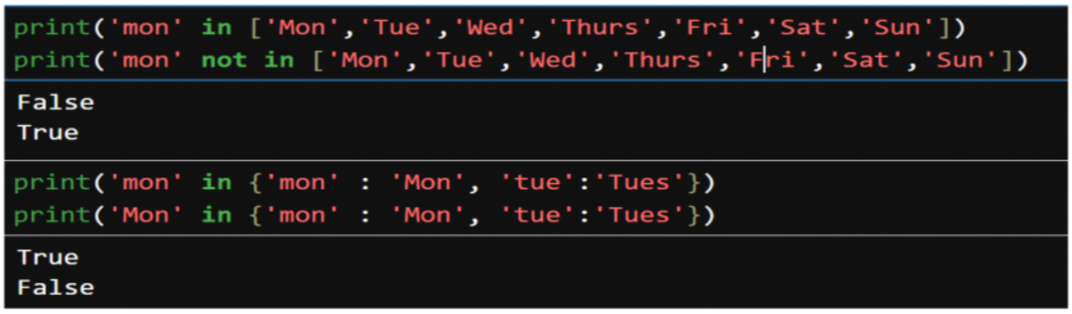
- Its Returns Boolean Value
- 'in' -> True if the value is in the sequences
- 'not in' -> True if the value is not in the sequence

```python
print('hello' in 'hello world')
print('hello' not in 'hellz world')
```

```
True
True
```

```python
print('I' in 'India')
print('I' not in 'America')
```

```
True
True
```

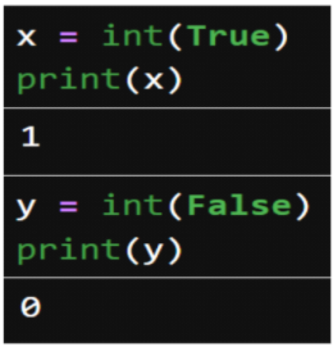


```python
print('mon' in ['Mon','Tue','Wed','Thurs','Fri','Sat','Sun'])
print('mon' not in ['Mon','Tue','Wed','Thurs','Fri','Sat','Sun'])
```

```
False
True
```

```python
print('mon' in {'mon' : 'Mon', 'tue':'Tues'})
print('Mon' in {'mon' : 'Mon', 'tue':'Tues'})
```

```
True
False
```

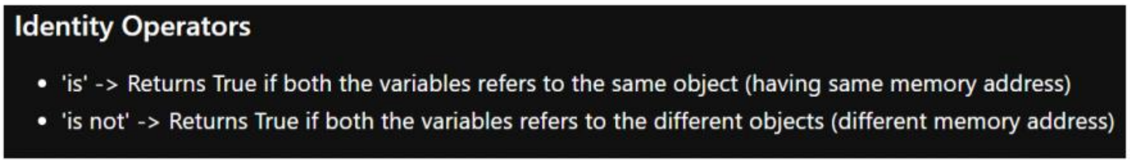


```python
x = int(True)
print(x)
```

```
1
```

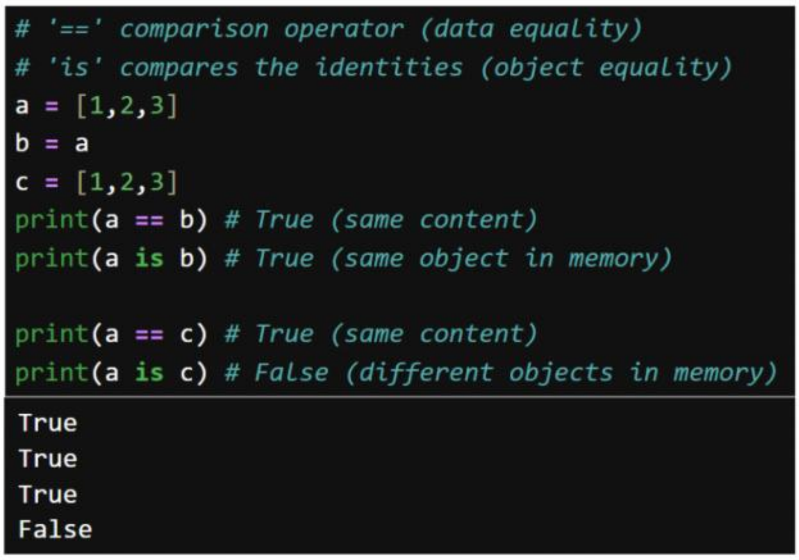
```python
y = int(False)
print(y)
```

```
0
```



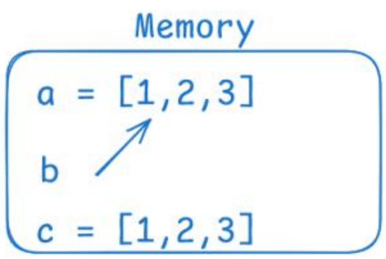
Identity Operators

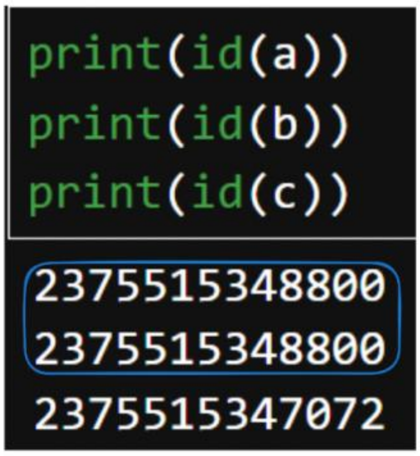
- 'is' -> Returns True if both the variables refers to the same object (having same memory address)
- 'is not' -> Returns True if both the variables refers to the different objects (different memory address)



```python
# '==' comparison operator (data equality)
# 'is' compares the identities (object equality)
a = [1,2,3]
b = a
c = [1,2,3]
print(a == b) # True (same content)
print(a is b) # True (same object in memory)

print(a == c) # True (same content)
print(a is c) # False (different objects in memory)
```

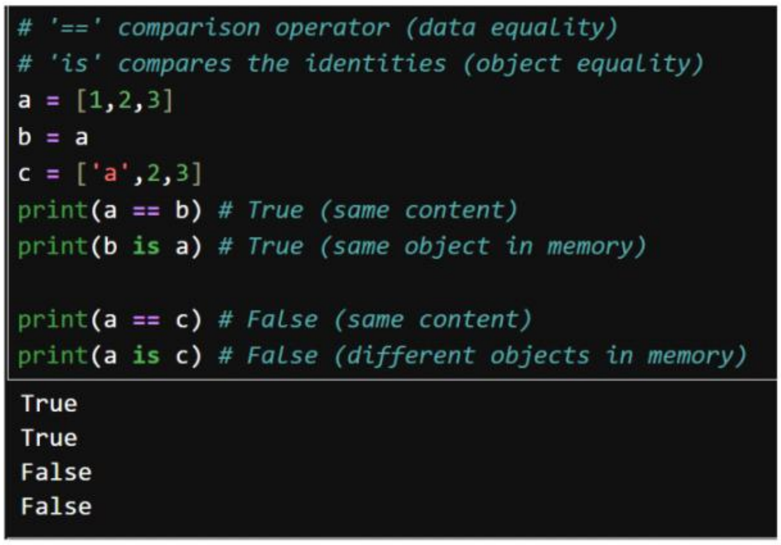
```
True
True
True
False
```





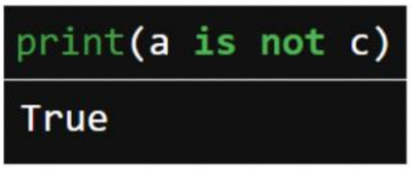
```python
print(id(a))
print(id(b))
print(id(c))
```

```
2375515348800
2375515348800
2375515347072
```



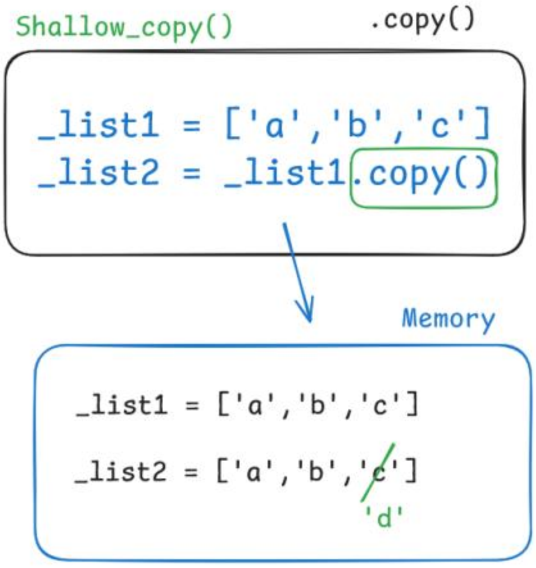
```python
# '==' comparison operator (data equality)
# 'is' compares the identities (object equality)
a = [1,2,3]
b = a
c = ['a',2,3]
print(a == b) # True (same content)
print(b is a) # True (same object in memory)

print(a == c) # False (same content)
print(a is c) # False (different objects in memory)
```

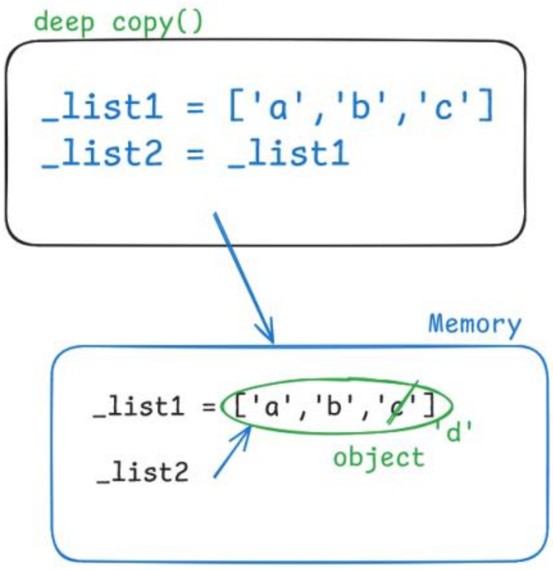
```
True
True
False
False
```



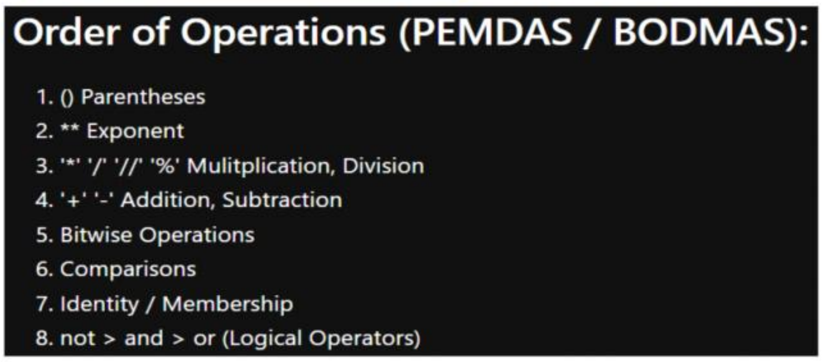
```python
print(a is not c)
```

```
True
```



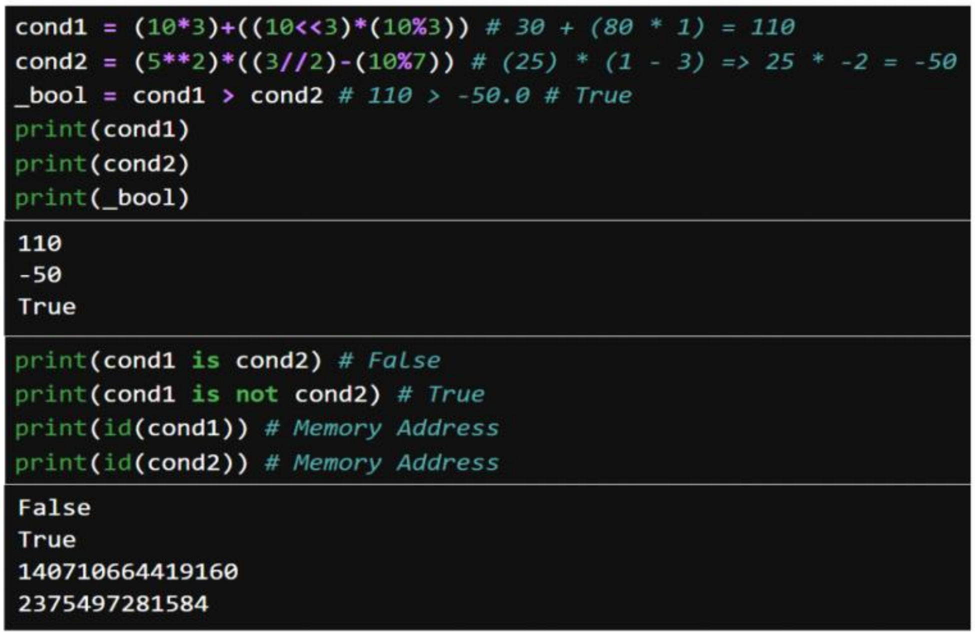


Orders Of Operations



Order of Operations (PEMDAS / BODMAS):

1. () Parentheses
2. ** Exponent
3. '*' '/' '//' '%' Mulitplication, Division
4. '+' '-' Addition, Subtraction
5. Bitwise Operations
6. Comparisons
7. Identity / Membership
8. not > and > or (Logical Operators)

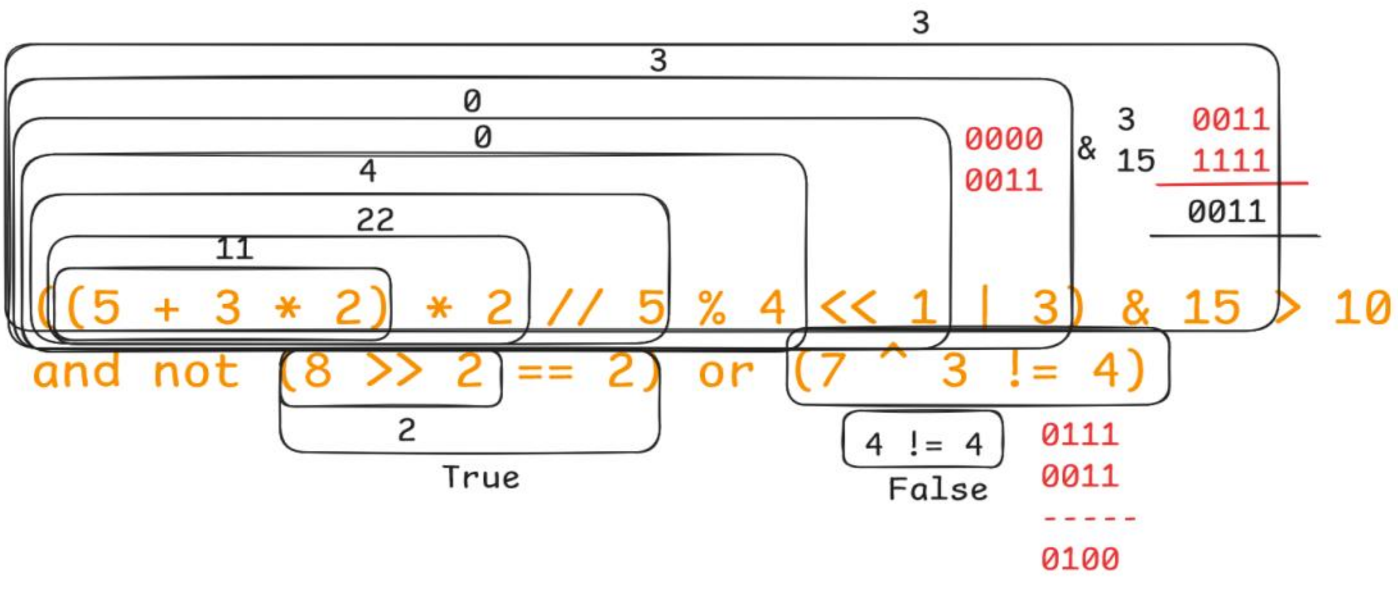


```python
cond1 = (10*3)+((10<<3)*(10%3)) # 30 + (80 * 1) = 110
cond2 = (5**2)*((3//2)-(10%7)) # (25) * (1 - 3) => 25 * -2 = -50
_bool = cond1 > cond2 # 110 > -50.0 # True
print(cond1)
print(cond2)
print(_bool)
```

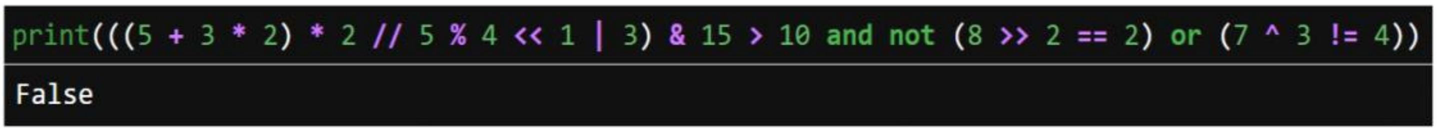
```
110
-50
True
```

```python
print(cond1 is cond2) # False
print(cond1 is not cond2) # True
print(id(cond1)) # Memory Address
print(id(cond2)) # Memory Address
```

```
False
True
140710664419160
2375497281584
```



```
False and not(True) or False
False and False or False => False
```

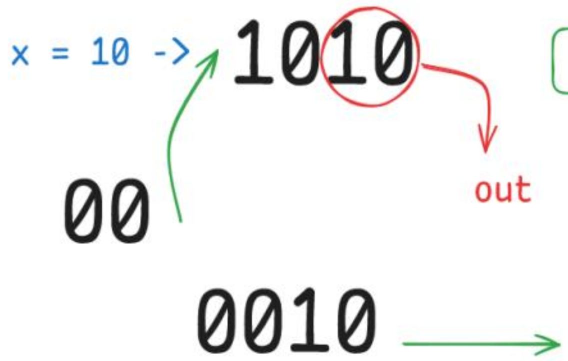


```python
print(((5 + 3 * 2) * 2 // 5 % 4 << 1 | 3) & 15 > 10 and not (8 >> 2 == 2) or (7 ^ 3 != 4))
```

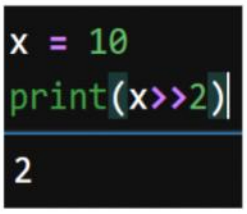
```
False
```

sir 8 >> 2 short me bta do??

Right Shift >>



x >> 2



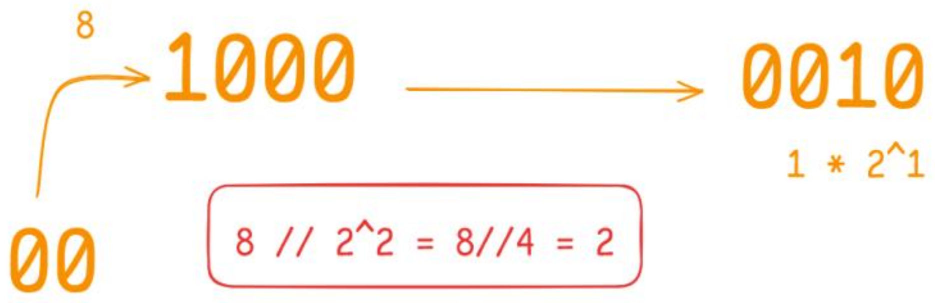
```python
x = 10
print(x>>2)
```

```
2
```

x // 2^shift

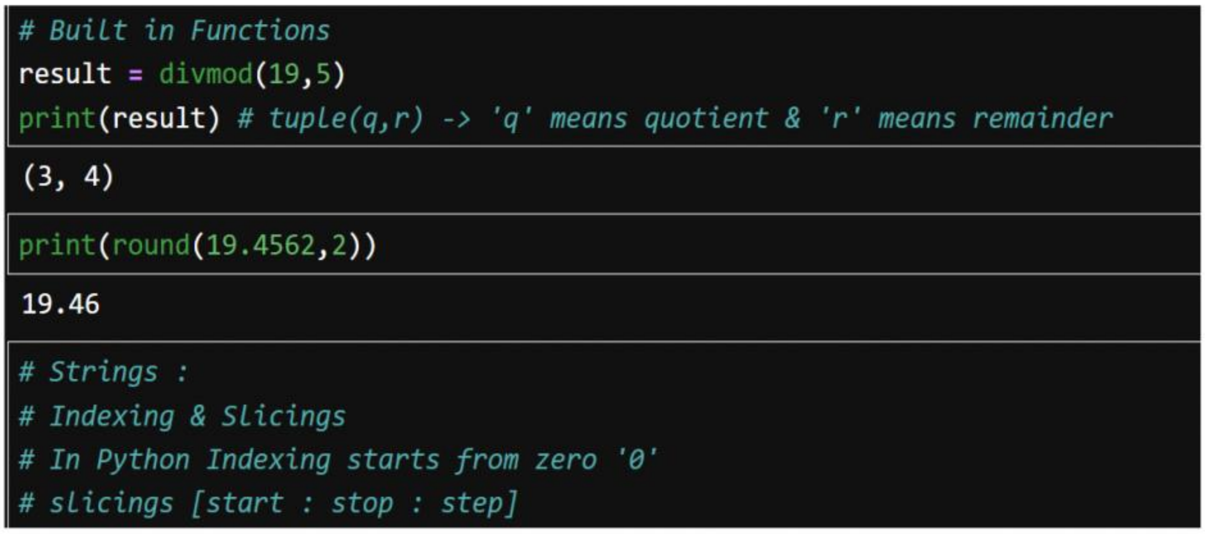
10 // 2^2 = 10 * 4 = 2

=(0 * 2^3) + (0 * 2^2) + (1 * 2^1) + (0 * 2^0)
= 0 + 0 + 2 + 0 = 2



1 * 2^1 = 2

8 // 2^2 = 8//4 = 2



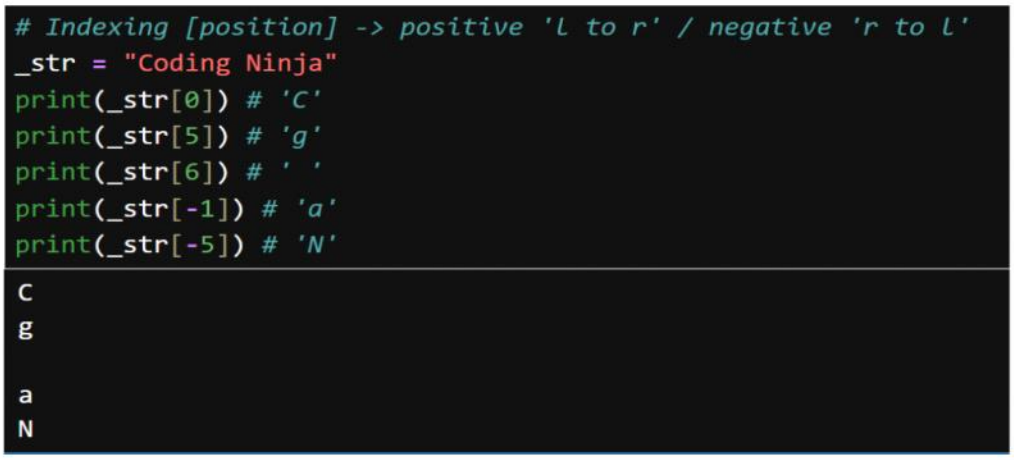
```python
# Built in Functions
result = divmod(19,5)
print(result) # tuple(q,r) -> 'q' means quotient & 'r' means remainder
```

```
(3, 4)
```

```python
print(round(19.4562,2))
```

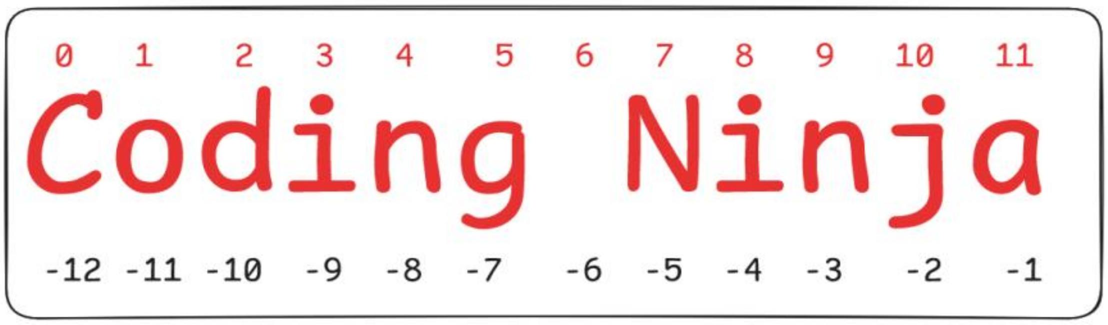
```
19.46
```

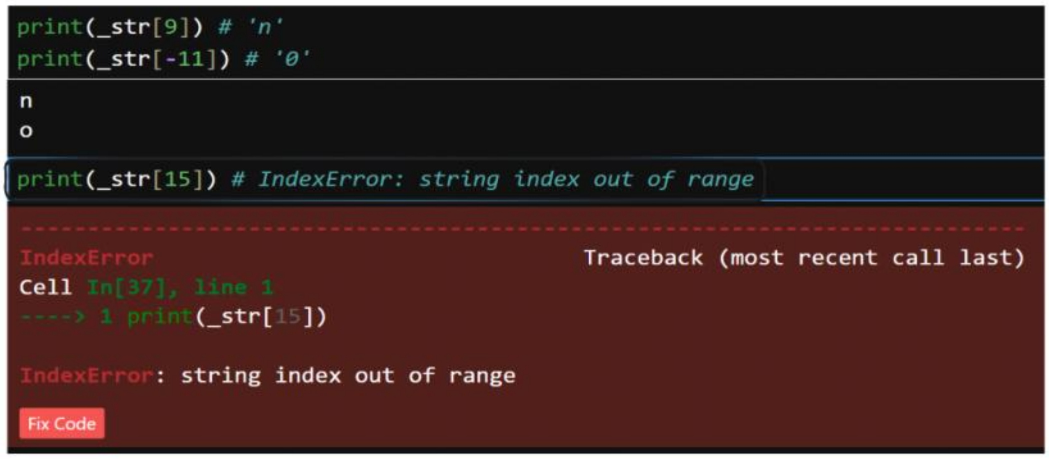
```python
# Strings :
# Indexing & Slicings
# In Python Indexing starts from zero '0'
# slicings [start : stop : step]
```



```
# Indexing [position] -> positive 'l to r' / negative 'r to l'
_str = "Coding Ninja"
print(_str[0]) # 'C'
print(_str[5]) # 'g'
print(_str[6]) # ' '
print(_str[-1]) # 'a'
print(_str[-5]) # 'N'
```

```
C
g

a
N
```





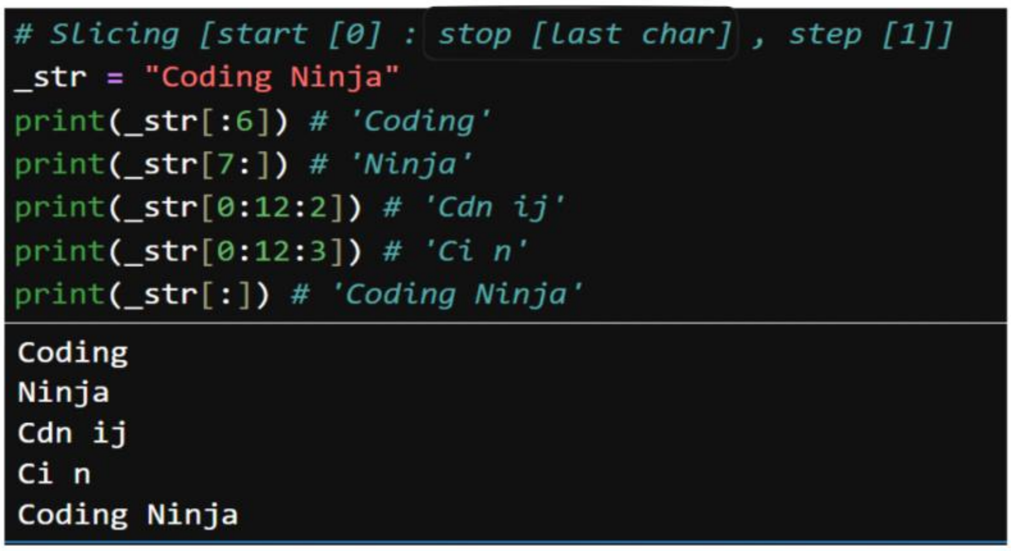
```
print(_str[9]) # 'n'
print(_str[-11]) # 'o'
```

```
n
o
```

```
print(_str[15]) # IndexError: string index out of range
```

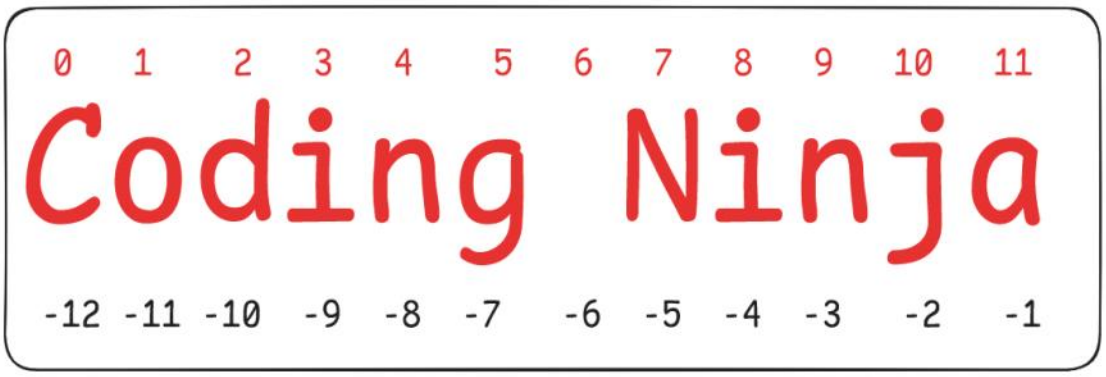
```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[37], line 1
----> 1 print(_str[15])

IndexError: string index out of range
```

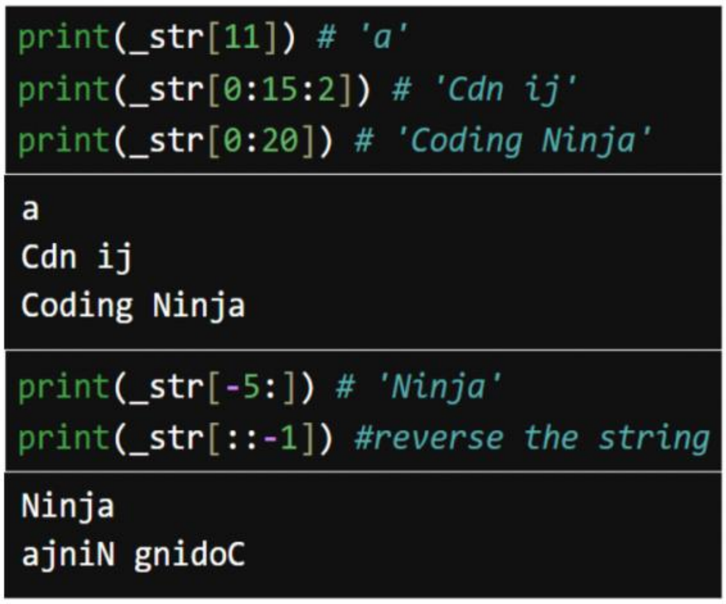
non-inclusive



```
# Slicing [start [0] : stop [last char] , step [1]]
_str = "Coding Ninja"
print(_str[:6]) # 'Coding'
print(_str[7:]) # 'Ninja'
print(_str[0:12:2]) # 'Cdn ij'
print(_str[0:12:3]) # 'Ci n'
print(_str[:]) # 'Coding Ninja'
```

```
Coding
Ninja
Cdn ij
Ci n
Coding Ninja
```



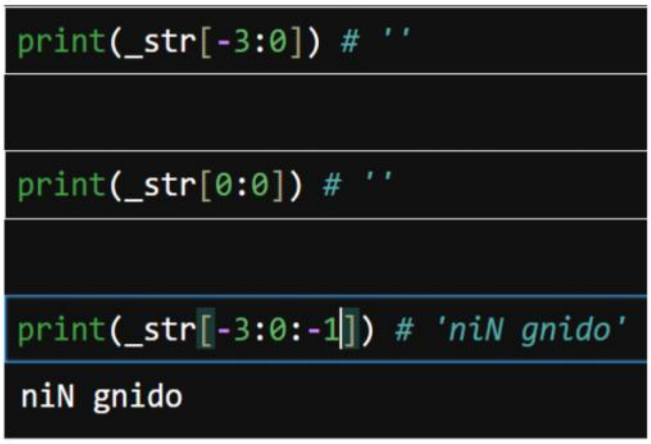


```
print(_str[11]) # 'a'
print(_str[0:15:2]) # 'Cdn ij'
print(_str[0:20]) # 'Coding Ninja'
```

```
a
Cdn ij
Coding Ninja
```

```
print(_str[-5:]) # 'Ninja'
print(_str[::-1]) #reverse the string
```

```
Ninja
ajniN gnidoC
```



```
print(_str[-3:0]) # ''
```

```
print(_str[0:0]) # ''
```

```
print(_str[-3:0:-1]) # 'niN gnido'
```

```
niN gnido
```