❖ Python Variables

```
number = 10
```

Assigning Values To Variables in Python

```
My_Name = 'Prasanjit Roy'

print(My_Name)
```

Changing The Value of a Variable in Python

```
My_Name = 'Prasanjit'

print(My_Name)

My_Name = ‘ROY’
print(My_Name)
```

Python Keywords

Keywords are predefined, reserved words used in Python programming that have special meanings to the compiler.

We cannot use a keyword as a variable name, function name, or any other identifier.

Python Keywords List				
False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Python Data Types

Python Data Types

Data Types	**Classes**	**Description**
Numeric	int, float, complex	Holds Numeric Values
String	str	Holds Sequence Of Characters
Sequence	list, tuple, range	Holds Collection Of Items

Mapping	dict	Holds Data In Key-Value Pair Form
Boolean	bool	Holds Either True Or False
Set	set, frozeenset	Hold Collection Of Unique Items

```
num1 = 5
print(num1, 'is of type', type(num1))

num2 = 2.0
print(num2, 'is of type', type(num2))

num3 = 1+2j
print(num3, 'is of type', type(num3))
```

```
name = 'Python'
print(name)

message = 'Python for beginners'
print(message)
```

Python Type Conversion

```python
integer_number = 123
float_number = 1.23

new_number = integer_number + float_number

# display new value and resulting data type
print("Value:",new_number)
print("Data Type:",type(new_number))
```

```python
num_string = '12'
num_integer = 23

print("Data type of num_string before Type Casting:",type(num_string))

# explicit type conversion
num_string = int(num_string)

print("Data type of num_string after Type Casting:",type(num_string))

num_sum = num_integer + num_string

print("Sum:",num_sum)
print("Data type of num_sum:",type(num_sum))
```

Python Basic Input and Output

```python
num = input('Enter a number: ')

print('You Entered:', num)
```

Print Concatenated Strings

```
print('Programiz is ' + 'awesome.')
```

Python Operators

Types of Python Operators

1. Arithmetic Operators
2. Assignment Operators
3. Comparison Operators
4. Logical Operators
5. Bitwise Operators
6. Special Operators

Operator	Operation	Example
`+`	Addition	`5 + 2 = 7`
`-`	Subtraction	`4 - 2 = 2`
`*`	Multiplication	`2 * 3 = 6`
`/`	Division	`4 / 2 = 2`
`//`	Floor Division	`10 // 3 = 3`
`%`	Modulo	`5 % 2 = 1`
`**`	Power	`4 ** 2 = 16`

Python Assignment Operators

Operator	Name	Example
`=`	Assignment Operator	`a = 7`
`+=`	Addition Assignment	`a += 1 # a = a + 1`
`-=`	Subtraction Assignment	`a -= 3 # a = a - 3`
`*=`	Multiplication Assignment	`a *= 4 # a = a * 4`
`/=`	Division Assignment	`a /= 3 # a = a / 3`
`%=`	Remainder Assignment	`a %= 10 # a = a % 10`
`**=`	Exponent Assignment	`a **= 10 # a = a ** 10`

Python Comparison Operators

Operator	Meaning	Example
`==`	Is Equal To	`3 == 5` gives us **False**
`!=`	Not Equal To	`3 != 5` gives us **True**
`>`	Greater Than	`3 > 5` gives us **False**
`<`	Less Than	`3 < 5` gives us **True**
`>=`	Greater Than or Equal To	`3 >= 5` give us **False**

<=	Less Than or Equal To	3 <= 5 gives us **True**

Python Logical Operators

Operator	Example	Meaning
and	a **and** b	**Logical AND**: True only if both the operands are True
or	a **or** b	**Logical OR**: True if at least one of the operands is True
not	**not** a	**Logical NOT**: True if the operand is False and vice-versa.

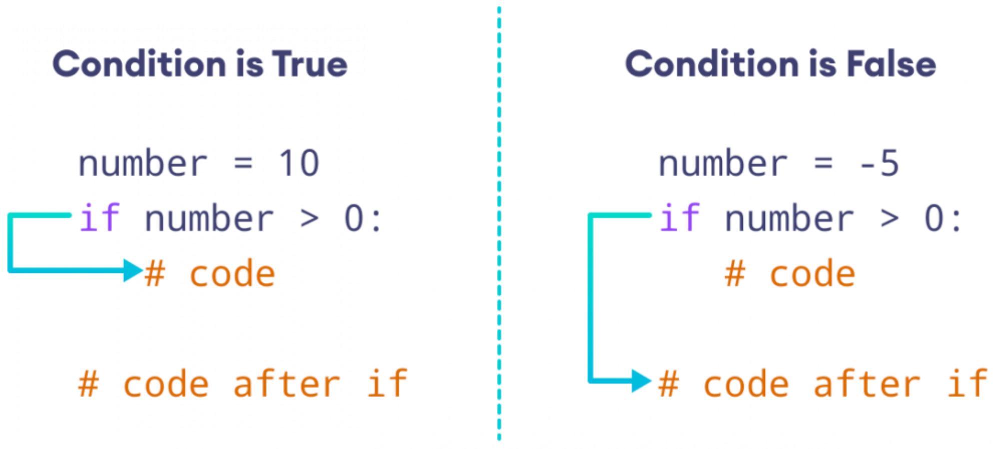
```
# logical AND
print(True and True)     # True
print(True and False)    # False

# logical OR
print(True or False)     # True

# logical NOT
print(not True)          # False
```

Python Bitwise Operators

Operator	Meaning	Example
&	Bitwise AND	x & y = 0 (0000 0000)

|	Bitwise OR	x | y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x >> 2 = 2 (0000 0010)
<<	Bitwise left shift	x << 2 = 40 (0010 1000)

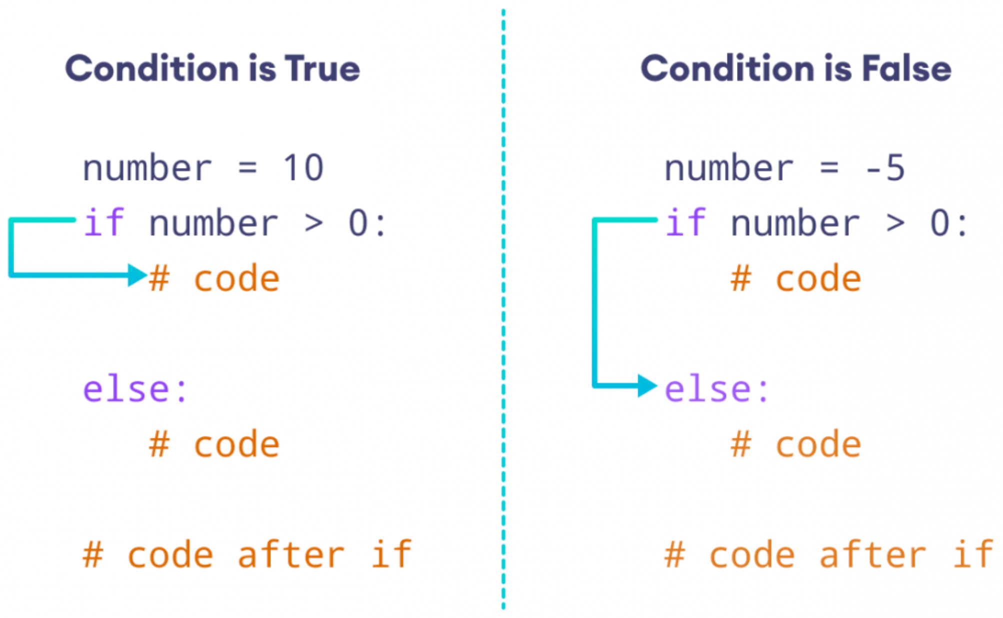
Python If...Else Statement



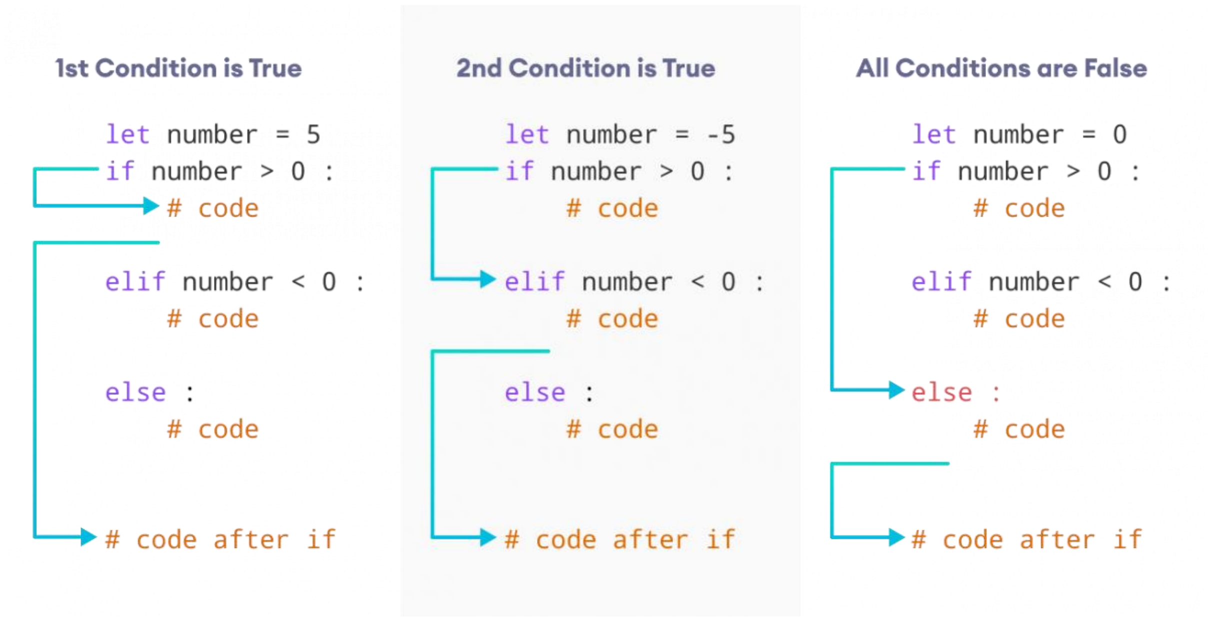
```
number = 10
if number > 0:
    # code

# code after if
```

```
number = -5
if number > 0:
    # code

# code after if
```

```
number = 10

# check if number is greater than 0
if number > 0:
   print('Number is positive.')

print('The if statement is easy')
```



```
number = 10

if number > 0:
   print('Positive number')

else:
   print('Negative number')

print('This statement is always executed')
```



```
let number = 5
if number > 0 :
    # code

elif number < 0 :
    # code

else :
    # code

# code after if
```

```
let number = -5
if number > 0 :
    # code

elif number < 0 :
    # code

else :
    # code

# code after if
```

```
let number = 0
if number > 0 :
    # code

elif number < 0 :
    # code

else :
    # code

# code after if
```

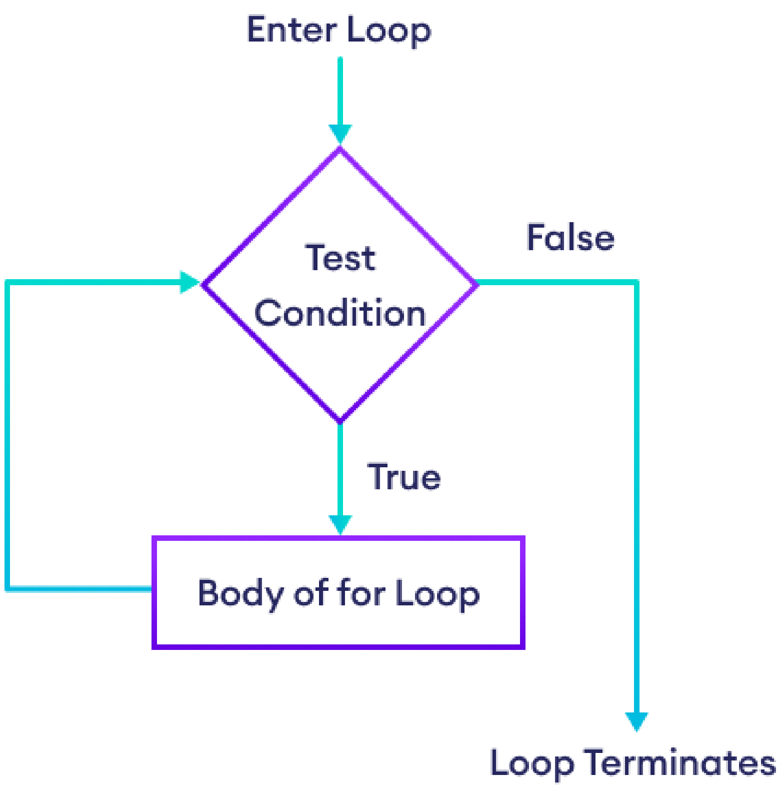
```
number = 0

if number > 0:
    print("Positive number")

elif number == 0:
    print('Zero')
else:
    print('Negative number')

print('This statement is always executed')
```

Python Nested if Statement

```
number = 5

# outer if statement
if (number >= 0):
    # inner if statement
    if number == 0:
      print('Number is 0')

    # inner else statement
    else:
        print('Number is positive')

# outer else statement
else:
    print('Number is negative')
```

Python For Loop

```
languages = ['Swift', 'Python', 'Go', 'JavaScript']

# run a loop for each item of the list
for language in languages:
    print(language)
```

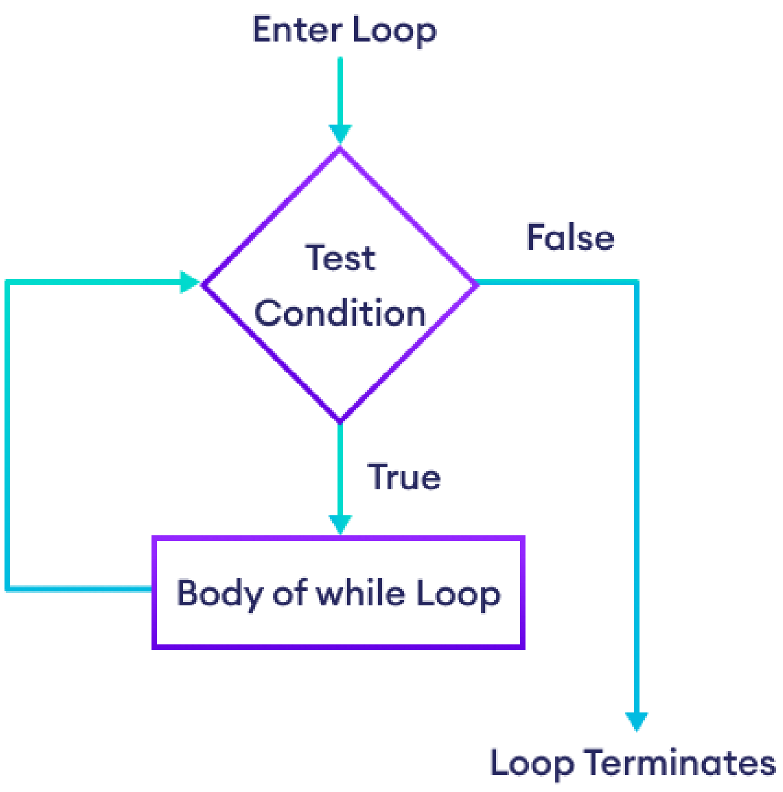


```
for x in 'Python':
   print(x)
```

Python for Loop with Python range()

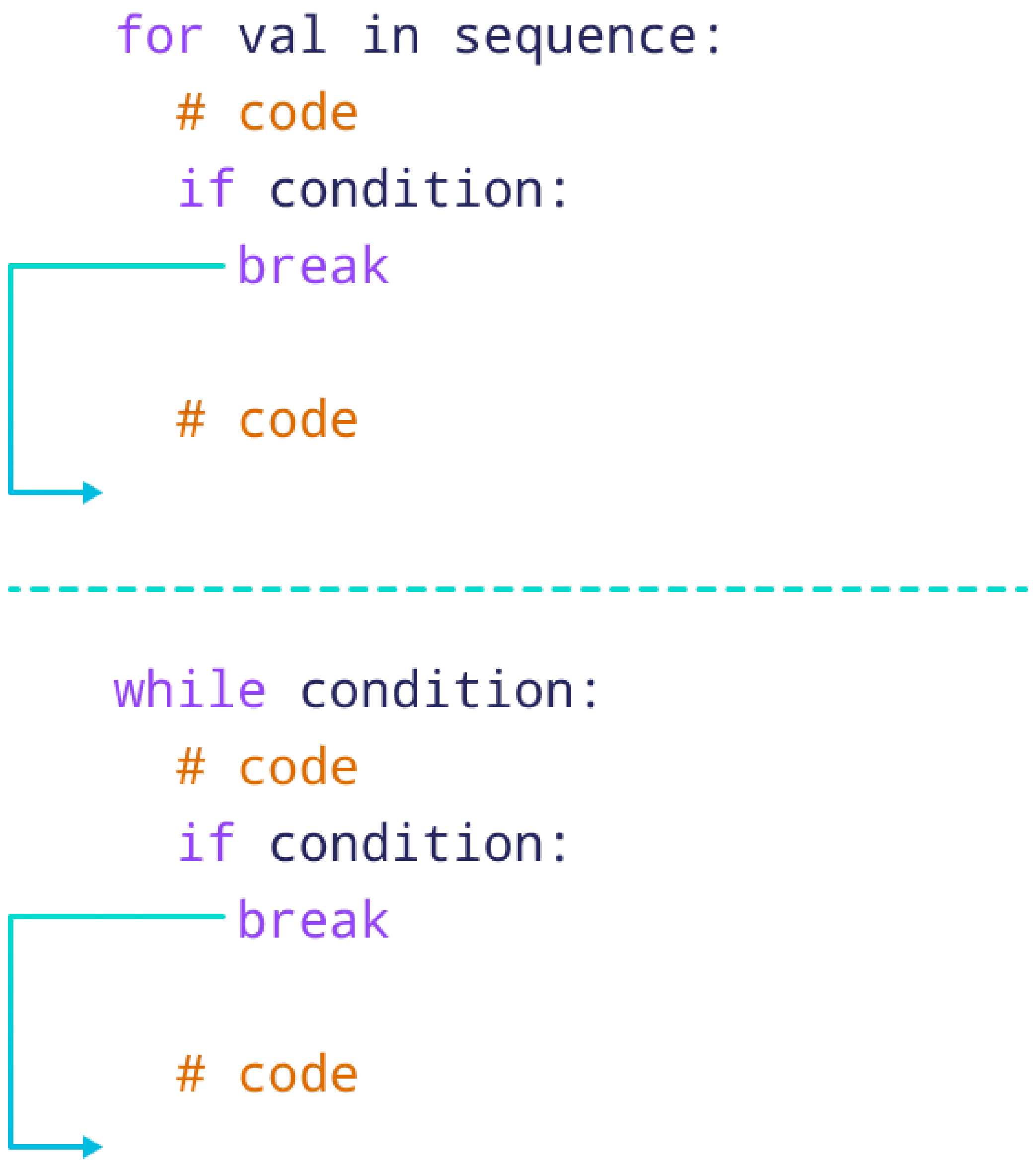
```
# use of range() to define a range of values
values = range(4)

# iterate from i = 0 to i = 3
for i in values:
   print(i)
```

Python while Loop



```
counter = 0

while counter < 3:
   print('Inside loop')
   counter = counter + 1
else:
   print('Inside else')
```

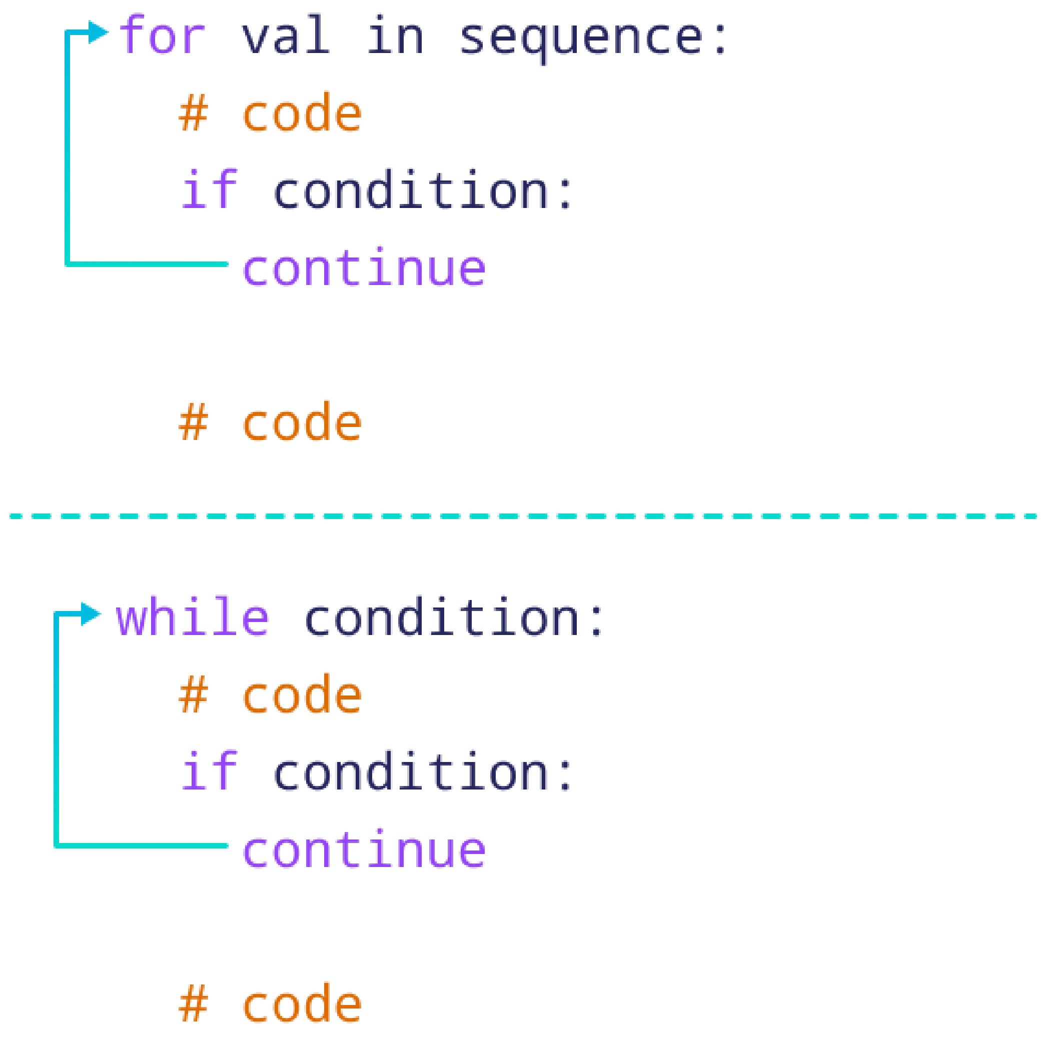
Python break and continue



```
for val in sequence:
  # code
  if condition:
    break

  # code
```

```
while condition:
  # code
  if condition:
    break

  # code
```

```
for i in range(5):
  if i == 3:
    break
  print(i)
```



```
for val in sequence:
  # code
  if condition:
    continue

  # code
```

```
while condition:
  # code
  if condition:
    continue

  # code
```

```
for i in range(5):
  if i == 3:
    continue
  print(i)
```