ikSaan.com

# Loops in Python

## Loops in Python

Loops in Python allow you to repeat a block of code multiple times. Python has two main types of loops: for loops and while loops. These loops are used to iterate over sequences (like lists, tuples, dictionaries, strings) or perform repeated tasks as long as a condition is true.

ikSaan.com

# Loops in Python

## for Loop

**The for loop in Python is used to iterate over a sequence (such as a list, tuple, dictionary, set, or string) or other iterable objects like ranges.**

```python
for item in iterable:
    # code to execute for each item
```

ikSaan.com

# Loops in Python

**Example:**

```python
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
# Output:
# apple
# banana
# cherry
```

ikSaan.com

# Loops in Python

## range() Function

**The range() function generates a sequence of numbers. It is commonly used with for loops when you want to iterate a specific number of times.**

```python
range(start, stop, step)
```

ikSaan.com

# Loops in Python

## range() Function

- start (optional): Starting number (default is 0)
- stop: Stopping number (not included)
- step (optional): Step size (default is 1)

```python
# Looping through a range of numbers
for i in range(5):
    print(i)
# Output: 0, 1, 2, 3, 4

# Looping with a start and end value
for i in range(2, 6):
    print(i)
# Output: 2, 3, 4, 5

# Looping with a step value
for i in range(0, 10, 2):
    print(i)
# Output: 0, 2, 4, 6, 8
```

ikSaan.com

# Loops in Python

## Iterating Over Strings

**A string is a sequence of characters, and you can use a for loop to iterate over each character.**

```python
for char in "Python":
    print(char)
# Output: P, y, t, h, o, n
```

ikSaan.com

# Loops in Python

## Iterating Over Strings

**A string is a sequence of characters, and you can use a for loop to iterate over each character.**

```python
for char in "Python":
    print(char)
# Output: P, y, t, h, o, n
```

ikSaan.com

# Loops in Python

## Iterating Over Dictionaries

**When you iterate over a dictionary, you can access its keys, values, or both.**

python

```python
# Iterating over keys
my_dict = {"name": "Alice", "age": 25}
for key in my_dict:
    print(key)
# Output: name, age

# Iterating over values
for value in my_dict.values():
    print(value)
# Output: Alice, 25

# Iterating over keys and values
for key, value in my_dict.items():
    print(f"{key}: {value}")
# Output: name: Alice, age: 25
```

ikSaan.com

# Loops in Python

## while Loop

**The while loop repeats a block of code as long as a specified condition is True. If the condition becomes False, the loop stops.**

```python
while condition:
    # code to execute while the condition is True
```

ikSaan.com

# Loops in Python

## while Loop

**Example:**

```python
x = 0
while x < 5:
    print(x)
    x += 1
# Output: 0, 1, 2, 3, 4
```

ikSaan.com

# Loops in Python

## Infinite Loops

**A while loop can become an infinite loop if the condition never becomes False.**

```python
while True:
    print("This is an infinite loop")
```

ikSaan.com

# break, continue, and pass Statements

## break Statement

**The break statement is used to exit a loop prematurely, even if the loop condition is still true.**

```python
for i in range(10):
    if i == 5:
        break  # Exit the loop when i is 5
    print(i)
# Output: 0, 1, 2, 3, 4
```

ikSaan.com

# break, continue, and pass Statements

## continue Statement

**The continue statement is used to skip the current iteration and move to the next iteration of the loop.**

**Example:**

```python
for i in range(5):
    if i == 3:
        continue  # Skip the iteration when i is 3
    print(i)
# Output: 0, 1, 2, 4
```

ikSaan.com

# break, continue, and pass Statements

## pass Statement

**The pass statement is a null operation—it does nothing. It's used as a placeholder where code is required but you don't want to execute any action.**

**Example:**

```python
for i in range(5):
    if i == 3:
        pass  # Placeholder, does nothing
    print(i)
# Output: 0, 1, 2, 3, 4
```

ikSaan.com

# break, continue, and pass Statements

## else with Loops

**The else clause in loops is executed after the loop finishes its normal iteration. It is not executed if the loop is exited using break.**

**Example:**

```python
for i in range(5):
    print(i)
else:
    print("Loop finished!")
# Output: 0, 1, 2, 3, 4, Loop finished!
```

ikSaan.com

# break, continue, and pass Statements

## Example with while loop:

**Example:**

```python
x = 0
while x < 5:
    print(x)
    x += 1
else:
    print("Loop finished!")
# Output: 0, 1, 2, 3, 4, Loop finished!
```

ikSaan.com

# Loops in Python

## Nested Loops

**You can have loops inside loops (nested loops). The inner loop completes all its iterations before the outer loop continues to the next iteration.**

**Example:**

python

```python
for i in range(3):
    for j in range(2):
        print(f"i = {i}, j = {j}")
# Output:
# i = 0, j = 0
# i = 0, j = 1
# i = 1, j = 0
# i = 1, j = 1
# i = 2, j = 0
# i = 2, j = 1
```

ikSaan.com

# Loops in Python

## Looping Through Lists with enumerate()

**The enumerate() function adds a counter to the loop, allowing you to access both the index and the value of each element during iteration.**

**Example:**

```python
fruits = ["apple", "banana", "cherry"]
for index, fruit in enumerate(fruits):
    print(f"Index: {index}, Fruit: {fruit}")
# Output:
# Index: 0, Fruit: apple
# Index: 1, Fruit: banana
# Index: 2, Fruit: cherry
```

ikSaan.com

# Loops in Python

## Looping Through Multiple Sequences with zip()

**The zip() function allows you to iterate over two or more sequences (like lists or tuples) simultaneously.**

**Example:**

```python
names = ["Alice", "Bob", "Charlie"]
ages = [25, 30, 35]

for name, age in zip(names, ages):
    print(f"{name} is {age} years old.")
# Output:
# Alice is 25 years old.
# Bob is 30 years old.
# Charlie is 35 years old.
```