Explain loops and different types in Python.

In Python, loops are used to execute a block of code repeatedly. There are two main types of loops:

1. **`for` loop**: This loop is used to iterate over a sequence (such as a list, tuple, string, or range) or other iterable objects. The loop continues until it has processed each item in the sequence.

   Python

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

2. **`while` loop**: This loop continues to execute a block of code as long as a specified condition is true.

   Python

   ```
   i = 1
   while i < 6:
       print(i)
       i += 1
   ```

Additional loop control statements:

- **`break`**: This statement terminates the loop and transfers execution to the statement immediately following the loop.

  Python

  ```
  fruits = ["apple", "banana", "cherry"]
  for fruit in fruits:
      if fruit == "banana":
          break
      print(fruit)
  ```

- **`continue`**: This statement skips the rest of the current iteration and jumps to the beginning of the next iteration.

  Python

  ```
  fruits = ["apple", "banana", "cherry"]
  for fruit in fruits:
      if fruit == "banana":
          continue
      print(fruit)
  ```

Nested loops:

Loops can be nested inside other loops, allowing for more complex iterations.

Python

```
for i in range(3):
    for j in range(2):
        print(i, j)
```

Looping with `range()` function:

The `range()` function is often used in loops to generate a sequence of numbers.

Python

```
for i in range(5):  # generates numbers from 0 to 4
    print(i)
```

Looping with `enumerate()` function:

The `enumerate()` function is used to iterate over a sequence while keeping track of the index of the current item.

Python

```
fruits = ["apple", "banana", "cherry"]
for index, fruit in enumerate(fruits):
    print(index, fruit)
```

These are the basic loop concepts in Python. Understanding these concepts will allow you to write efficient and effective code for various programming tasks.