In [16]:
```python
# Input: An integer number
num = int(input("enter value for fact"))

# Initialize the factorial variable to 1
factorial = 1

# Calculate the factorial using a for loop
for i in range(1, num + 1):
    factorial *= i

# Output: The factorial of the number
print(f"The factorial of {num} is {factorial}")
#In Python, the f (or F) prefix before a string literal denotes an f-string, which
```

```
The factorial of 5 is 120
```

In [6]:
```python
#only if condition
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

```
b is greater than a
```

In [7]:
```python
#if-else condition
i = 20

 # Checking if i is greater than 0
if (i > 0):
    print("i is positive")
else:
    print("i is 0 or Negative")
```

```
i is positive
```

In [ ]:
```python
#IF age is more than 30 years and experience is more than 10 years then candidate i
```

In [9]:
```python
i = 1
while i < 6:
  print(i)
  i += 1
```

```
1
2
3
4
5
```

In [3]:
```python
# a list of three elements
ages = [19, 26, 29]
print(ages)
```

```
[19, 26, 29]
```

In [9]:
```python
list1=[1,2,3,4,5,5,3,2,'A','A','B']
print(list1)
```

```
[1, 2, 3, 4, 5, 5, 3, 2, 'A', 'A', 'B']
```

In [6]:
```
#List Characteristics
#In Python, lists are:

#Ordered - They maintain the order of elements.
#Mutable - Items can be changed after creation.
#Allow duplicates - They can contain duplicate values.
```

In [ ]:
```
#Access List Elements
#Each element in a list is associated with a number, known as an index. The index o
languages = ['Python', 'Java', 'C++']

# access the first element
print('languages[0] =', languages[0])

# access the third element
print('languages[2] =', languages[2])
```

In [ ]:
```
#Negative Indexing
#In Python, a list can also have negative indices. The index of the last element is
languages = ['Python', 'Java', 'C++']

# access the last item
print('languages[-1] =', languages[-1])

# access the third last item
print('languages[-3] =', languages[-3])
```

In [ ]:
```
#Slicing of a List in Python
#If we need to access a portion of a list, we can use the slicing operator, :
```

In [2]:
```
my_list = ['p', 'r', 'o', 'g', 'r', 'a', 'm']
print("my_list =", my_list)

# get a list with items from index 2 to index 4 (index 5 is not included)
print("my_list[2: 5] =", my_list[2: 5])

# get a list with items from index 2 to index -3 (index -2 is not included)
print("my_list[2: -2] =", my_list[2: -2])

# get a list with items from index 0 to index 2 (index 3 is not included)
print("my_list[0: 3] =", my_list[0: 3])
```

```
my_list = ['p', 'r', 'o', 'g', 'r', 'a', 'm']
my_list[2: 5] = ['o', 'g', 'r']
my_list[2: -2] = ['o', 'g', 'r']
my_list[0: 3] = ['p', 'r', 'o']
```

In [ ]:
```
#Omitting Start and End Indices in Slicing - If you omit the start index, the slici
```

In [3]:
```
my_list = ['p', 'r', 'o', 'g', 'r', 'a', 'm']
print("my_list =", my_list)

# get a list with items from index 5 to last
```

```
print("my_list[5: ] =", my_list[5: ])

# get a list from the first item to index -5
print("my_list[: -4] =", my_list[: -4])

# omitting both start and end index
# get a list from start to end items
print("my_list[:] =", my_list[:])
```

```
my_list = ['p', 'r', 'o', 'g', 'r', 'a', 'm']
my_list[5: ] = ['a', 'm']
my_list[: -4] = ['p', 'r', 'o']
my_list[:] = ['p', 'r', 'o', 'g', 'r', 'a', 'm']
```

In [ ]:
```
#Add Elements to a Python List - As mentioned earlier, lists are mutable and we can
```

In [6]:
```
fruits = ['apple', 'banana', 'orange']
print('Original List:', fruits)

fruits.append('cherry')

print('Updated List:', fruits)
```

```
Original List: ['apple', 'banana', 'orange']
Updated List: ['apple', 'banana', 'orange', 'cherry']
```

In [ ]:
```
#Add Elements at the Specified Index We can insert an element at the specified inde
```

In [5]:
```
fruits = ['apple', 'banana', 'orange']
print("Original List:", fruits)

fruits.insert(2, 'cherry')

print("Updated List:", fruits)
```

```
Original List: ['apple', 'banana', 'orange']
Updated List: ['apple', 'banana', 'cherry', 'orange']
```

Python List Methods Python has many useful list methods that make it really easy to work with lists.

Method Description append() Adds an item to the end of the list extend() Adds items of lists and other iterables to the end of the list insert() Inserts an item at the specified index remove() Removes the specified value from the list pop() Returns and removes item present at the given index clear() Removes all items from the list index() Returns the index of the first matched item count() Returns the count of the specified item in the list sort() Sorts the list in ascending/descending order reverse() Reverses the item of the list copy() Returns the shallow copy of the list

Python Dictionary A Python dictionary is a collection of items, similar to lists and tuples. However, unlike lists and tuples, each item in a dictionary is a key-value pair (consisting of a key and a value).

In [7]:
```
# creating a dictionary
country_capitals = {
  "Germany": "Berlin",
```

```
  "Canada": "Ottawa",
  "England": "London"
}

# printing the dictionary
print(country_capitals)
```

```
{'Germany': 'Berlin', 'Canada': 'Ottawa', 'England': 'London'}
```

In [8]:
```
#Access Dictionary Items
#We can access the value of a dictionary item by placing the key inside square brac
country_capitals = {
  "Germany": "Berlin",
  "Canada": "Ottawa",
  "England": "London"
}
# access the value of keys
print(country_capitals["Germany"])    # Output: Berlin
print(country_capitals["England"])    # Output: London
```

```
Berlin
London
```

In [ ]:
```
#Add Items to a Dictionary
#We can add an item to a dictionary by assigning a value to a new key. For example,
country_capitals = {
  "Germany": "Berlin",
  "Canada": "Ottawa",
}

# add an item with "Italy" as key and "Rome" as its value
country_capitals["Italy"] = "Rome"

print(country_capitals)
```

Iterate Through a Dictionary A dictionary is an ordered collection of items (starting from Python 3.7), therefore it maintains the order of its items. We can iterate through dictionary keys one by one using a for loop.

In [1]:
```
country_capitals = {
  "United States": "Washington D.C.",
  "Italy": "Rome"
}

# print dictionary keys one by one
for country in country_capitals:
    print(country)

print()

# print dictionary values one by one
for country in country_capitals:
    capital = country_capitals[country]
    print(capital)
```

```
United States
Italy

Washington D.C.
Rome
```

Python Dictionary Methods Here are some of the commonly used dictionary methods.

Function Description pop() Removes the item with the specified key. update() Adds or changes dictionary items. clear() Remove all the items from the dictionary. keys() Returns all the dictionary's keys. values() Returns all the dictionary's values. get() Returns the value of the specified key. popitem() Returns the last inserted key and value as a tuple. copy() Returns a copy of the dictionary.