set operations can be performed using built-in set methods or operators. Here are the main types of set operations in Python:

1. UNION
Combines elements from both sets, removing duplicates.

```
A = {1, 2, 3}
B = {3, 4, 5}
union_set = A | B
print(union_set)  # Output: {1, 2, 3, 4, 5}
```

2. INTERSECTION
Finds common elements between two sets.

```
A = {1, 2, 3}
B = {3, 4, 5}
intersection_set = A & B
print(intersection_set)  # Output: {3}
```

3. DIFFERENCE
Finds elements in one set but not in another.

```
A = {1, 2, 3}
B = {3, 4, 5}
difference_set = A - B
print(difference_set)  # Output: {1, 2}
```

CONDITIONAL STATEMENTS
Conditional statements allow a program to execute different blocks of code based on conditions. Python provides several conditional statements:

1. IF STATEMENT
Executes a block of code if the condition is True.

```
age = 18
if age >= 18:
    print("You are an adult.")  # Output: You are an adult.
```

2. IF-ELSE STATEMENT
Provides an alternative block of code if the condition is False.

```
age = 16
if age >= 18:
    print("You are an adult.")
else:
    print("You are a minor.")  # Output: You are a minor.
```

3. IF-ELIF-ELSE STATEMENT
Allows multiple conditions to be checked in sequence.

```
age = 10
if age >= 18:
    print("You are an adult.")
elif age >= 13:
    print("You are a teenager.")
else:
    print("You are a child.")  # Output: You are a child.
```

4. NESTED IF STATEMENTS
An if statement inside another if.

```
age = 20
has_id = True
if age >= 18:
    if has_id:
        print("You can enter.")
    else:
        print("You need an ID.")
```

```
else:
    print("You are too young.")

5. TERNARY CONDITIONAL (if in One Line)
A shorthand way to write an if-else statement.

age = 20
status = "Adult" if age >= 18 else "Minor"

print(status)  # Output: Adult

     LOOPS
Loops are used to execute a block of code multiple times. And there are two main types
of loops:

1. FOR LOOP
A for loop is used to iterate over a sequence (like a list, tuple, dictionary, string,
or range). It executes a block of code for each element in the sequence.

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

 * Use for loops to iterate over sequences.
 * range() helps generate number sequences.
 * break stops the loop, and continue skips an iteration.
 * else runs after a normal loop execution.

2. WHILE LOOP
A while loop repeatedly executes a block of code as long as the given condition is True.
It is useful when the number of iterations is not known beforehand.

EXAMPLE:
count = 0
while count < 5:
    print(count)
    count ++

```