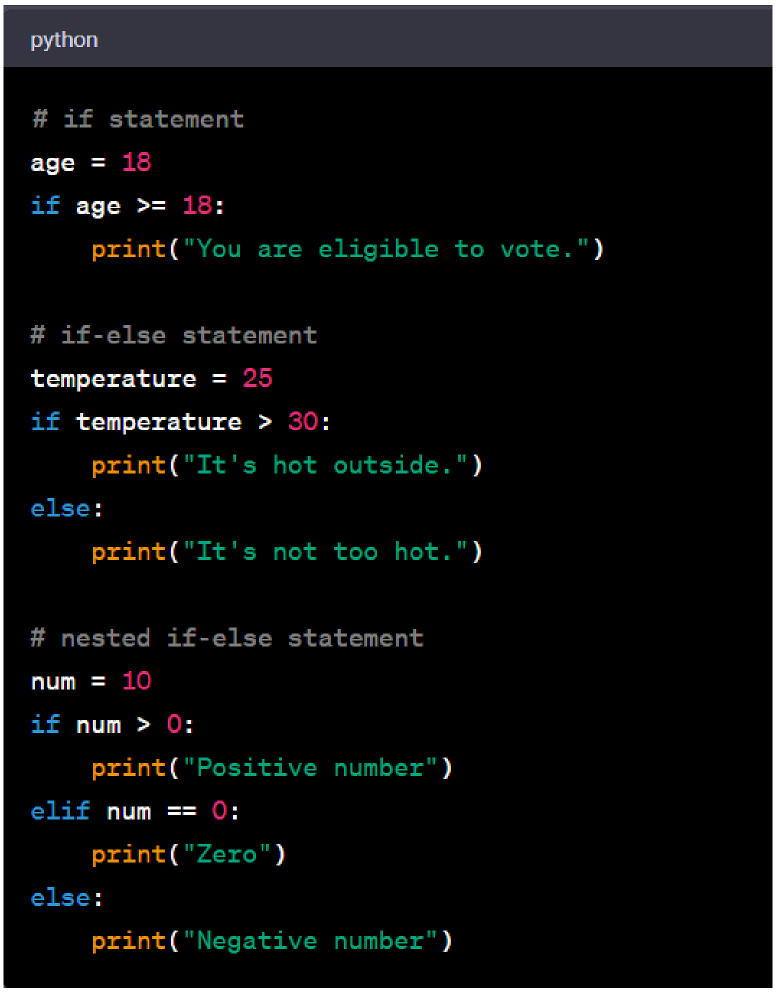
Control Statements

Control statements are programming language constructs that enable the control flow of a program, allowing you to alter the execution sequence based on certain conditions or loops. The most common control statements found in programming languages are:

1. Conditional Statements:

- if statement: Executes a block of code if a specified condition is true.
- if-else statement: Executes one block of code if a specified condition is true and another block if it is false.
- nested if-else statement: Contains if-else statements within other if-else statements.
- Python:



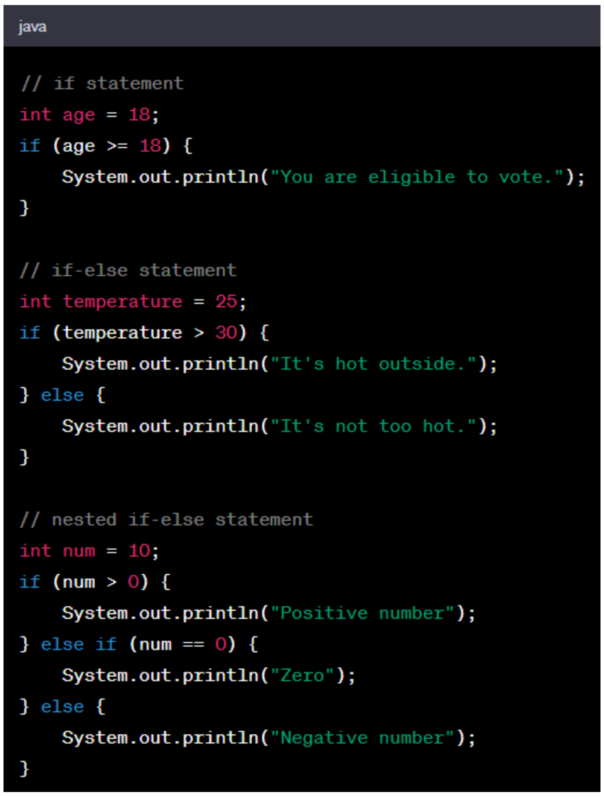
python

```python
# if statement
age = 18
if age >= 18:
    print("You are eligible to vote.")

# if-else statement
temperature = 25
if temperature > 30:
    print("It's hot outside.")
else:
    print("It's not too hot.")

# nested if-else statement
num = 10
if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
```

- Java:



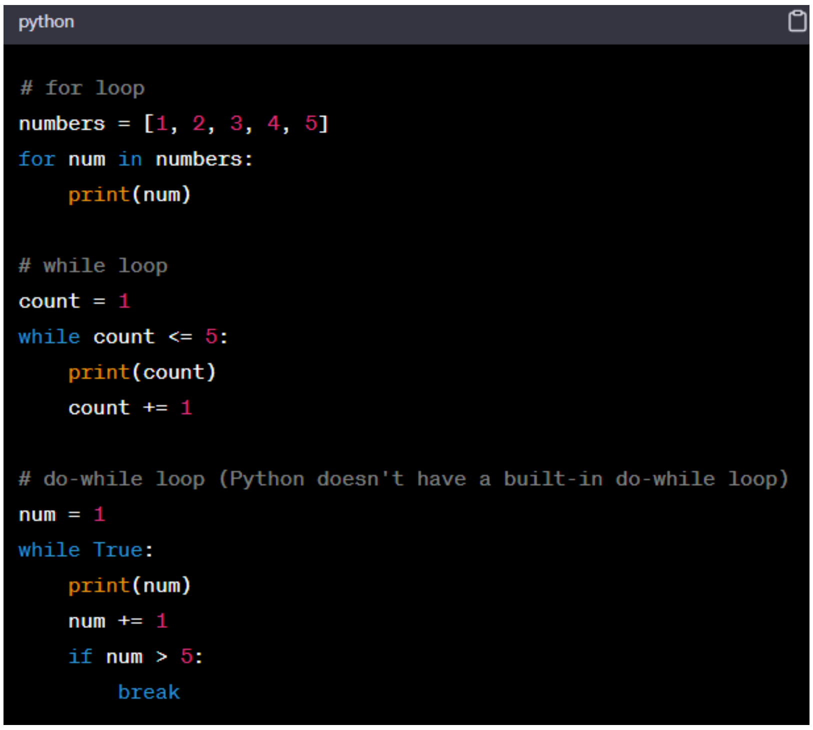
```java
// if statement
int age = 18;
if (age >= 18) {
    System.out.println("You are eligible to vote.");
}

// if-else statement
int temperature = 25;
if (temperature > 30) {
    System.out.println("It's hot outside.");
} else {
    System.out.println("It's not too hot.");
}

// nested if-else statement
int num = 10;
if (num > 0) {
    System.out.println("Positive number");
} else if (num == 0) {
    System.out.println("Zero");
} else {
    System.out.println("Negative number");
}
```

2. Looping Statements:

- for loop: Executes a block of code for a specific number of times, iterating over a range or collection of values.
- while loop: Repeatedly executes a block of code as long as a specified condition is true.
- do-while loop: Executes a block of code at least once and then repeatedly executes it as long as a specified condition is true.
- Python:



```python
# for loop
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    print(num)

# while loop
count = 1
while count <= 5:
    print(count)
    count += 1

# do-while loop (Python doesn't have a built-in do-while loop)
num = 1
while True:
    print(num)
    num += 1
    if num > 5:
        break
```

- Java:

java

```java
// for loop
int[] numbers = {1, 2, 3, 4, 5};
for (int num : numbers) {
    System.out.println(num);
}

// while loop
int count = 1;
while (count <= 5) {
    System.out.println(count);
    count++;
}

// do-while loop
int num = 1;
do {
    System.out.println(num);
    num++;
} while (num <= 5);
```

3. Switch Statement (or Case Statement):

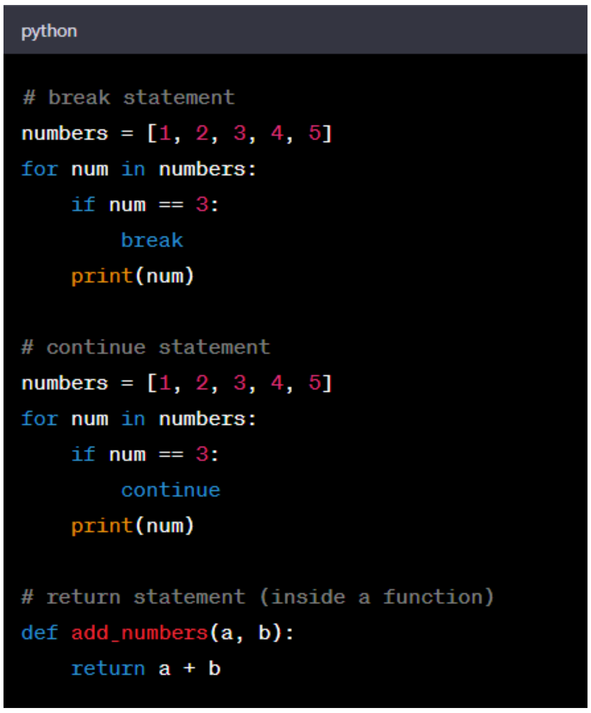
- switch statement: Evaluates an expression and executes code blocks based on different cases or values.
- Python: Python doesn't have a direct switch statement. You can achieve similar functionality using if-elif-else statements.
- Java:

java

```java
int day = 3;
switch (day) {
    case 1:
        System.out.println("Monday");
        break;
    case 2:
        System.out.println("Tuesday");
        break;
    case 3:
        System.out.println("Wednesday");
        break;
    default:
        System.out.println("Invalid day");
}
```

4. Jump Statements:

- break statement: Terminates the execution of a loop or switch statement and transfers control to the next statement outside the block.
- continue statement: Skips the current iteration of a loop and continues with the next iteration.
- return statement: Terminates the execution of a function and returns a value to the caller.
- Python:

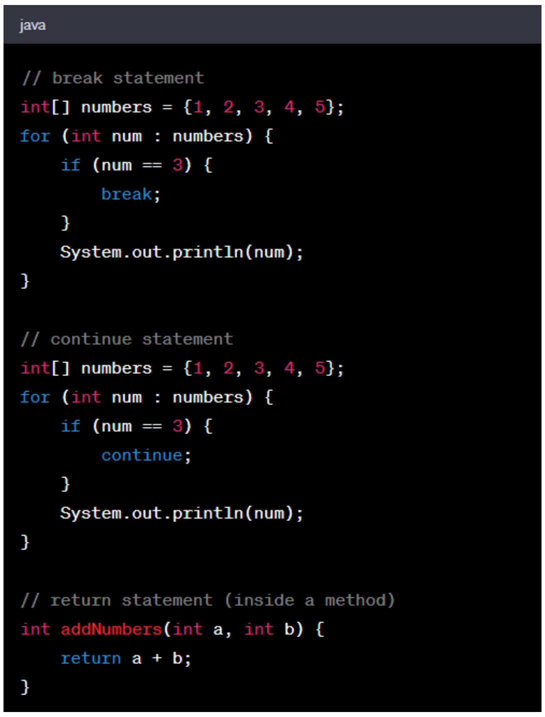


```python
# break statement
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    if num == 3:
        break
    print(num)

# continue statement
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    if num == 3:
        continue
    print(num)

# return statement (inside a function)
def add_numbers(a, b):
    return a + b
```

Java:



```java
// break statement
int[] numbers = {1, 2, 3, 4, 5};
for (int num : numbers) {
    if (num == 3) {
        break;
    }
    System.out.println(num);
}

// continue statement
int[] numbers = {1, 2, 3, 4, 5};
for (int num : numbers) {
    if (num == 3) {
        continue;
    }
    System.out.println(num);
}

// return statement (inside a method)
int addNumbers(int a, int b) {
    return a + b;
}
```

These control statements provide programmers with the ability to create more dynamic and flexible programs by controlling the flow of execution based on specific conditions, loops, or jumps. However, the availability and syntax of control statements can vary between programming languages.