Python Flow Control

- if...else
- for Loop
- while loop
- Break and continue
- Pass statement

if statement

- In computer programming, the **if statement** is a conditional statement.
- It is used to execute a block of code only when a specific condition is met.

  For example,

  Suppose we need to assign different grades to students based on their scores.

1. If a student scores above 90, assign grade A
2. If a student scores above 75, assign grade B
3. If a student scores above 65, assign grade C

   These conditional tasks can be achieved using the if statement.

An if statement executes a block of code only when the specified condition is met.

Syntax

```
if condition:
    # body of if statement
```

- If condition evaluates to **True**, the body of the if statement is executed.
- If condition evaluates to **False**, the body of the if statement will be skipped from execution.

Condition is True

```
number = 10
if number > 0:
    # code

# code after if
```

Condition is False

```
number = -5
if number > 0:
    # code

# code after if
```

- If user enters 10, the condition number > 0 evaluates to True. Therefore, the body of if is executed.
- If user enters -5, the condition number > 0 evaluates to False. Therefore, the body of if is skipped from execution

Indentation in Python

- **Python uses indentation to define a block of code, such as the body of an if statement.**

```
x = 1
total = 0
# start of the if statement
if x != 0:
    total += x
    print(total)
# end of the if statement
```

Here, the body of if has two statements. We know this because two statements (immediately after if) start with indentation. We usually use four spaces for indentation in Python, although any number of spaces works as long as we are consistent.

You will get an error if you write the above code like this:

```
# Error code
x = 1
total = 0

if x != 0:
total += x
print(total)
```

Python if...else Statement

An if statement can have an optional else clause. The else statement executes if the condition in the if statement evaluates to False.

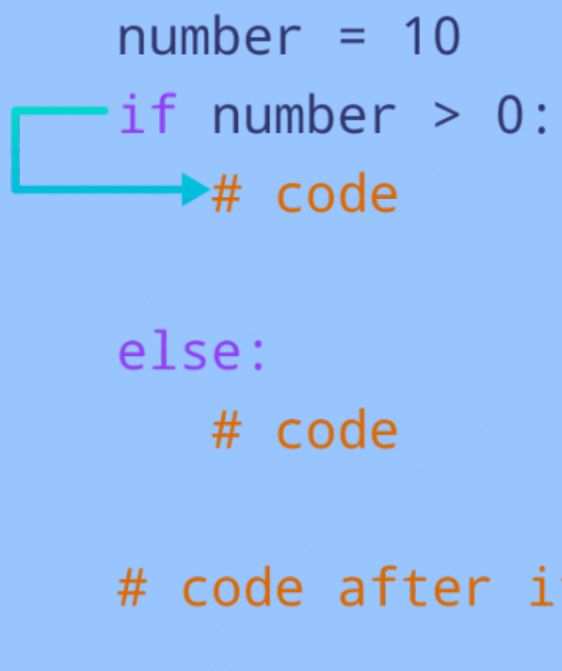
Syntax

```
if condition:
    # body of if statement
else:
    # body of else statement
```

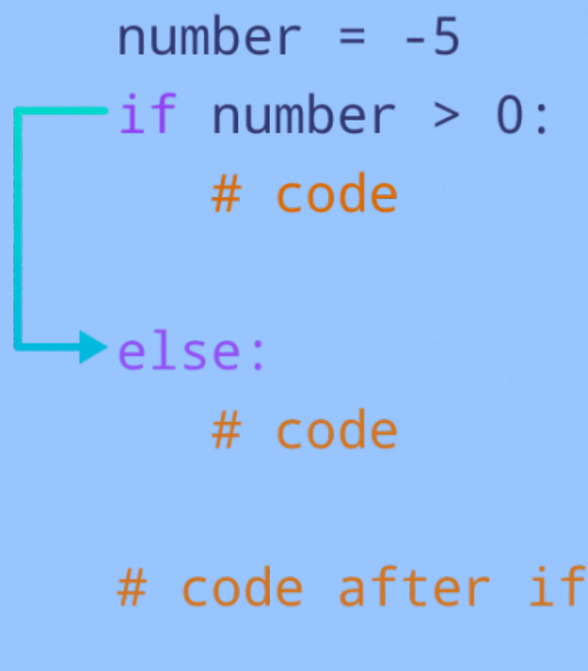
Here, if the condition inside the if statement evaluates to

- **True** - the body of if executes, and the body of else is skipped.
- **False** - the body of else executes, and the body of if is skipped

Condition is True



Condition is False



if...elif...else Statement

- The if...else statement is used to execute a block of code among two alternatives.
- However, if we need to make a choice between more than two alternatives, we use the if...elif...else statement.

Syntax

```
if condition1:
    # code block 1
elif condition2:
    # code block 2
else:
    # code block 3
```

1st Condition is True

```
let number = 5
if number > 0 :
    # code

elif number < 0 :
    # code

else :
    # code


# code after if
```

2nd Condition is True

```
let number = -5
if number > 0 :
    # code

elif number < 0 :
    # code

else :
    # code


# code after if
```

All Conditions are False

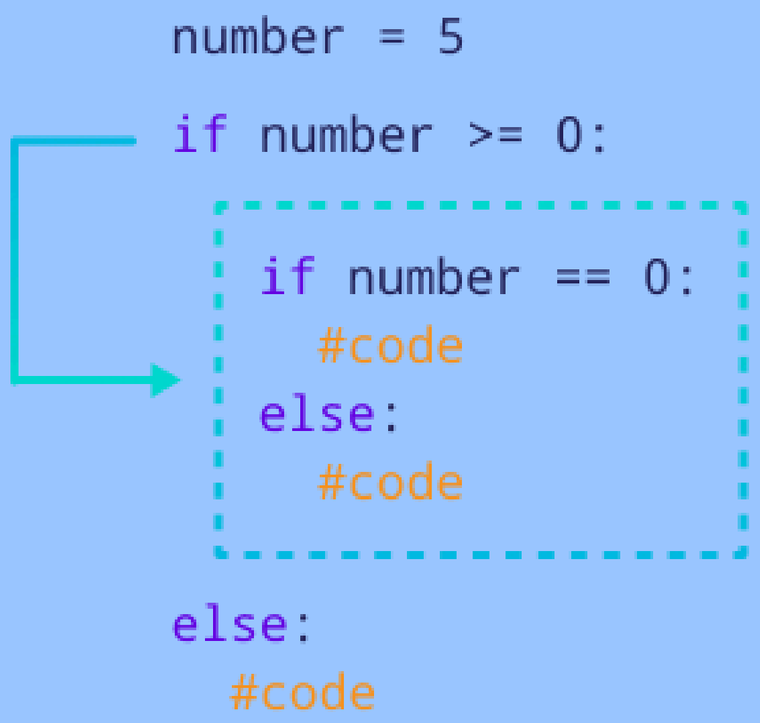
```
let number = 0
if number > 0 :
    # code

elif number < 0 :
    # code

else :
    # code


# code after if
```

Python Nested if statements

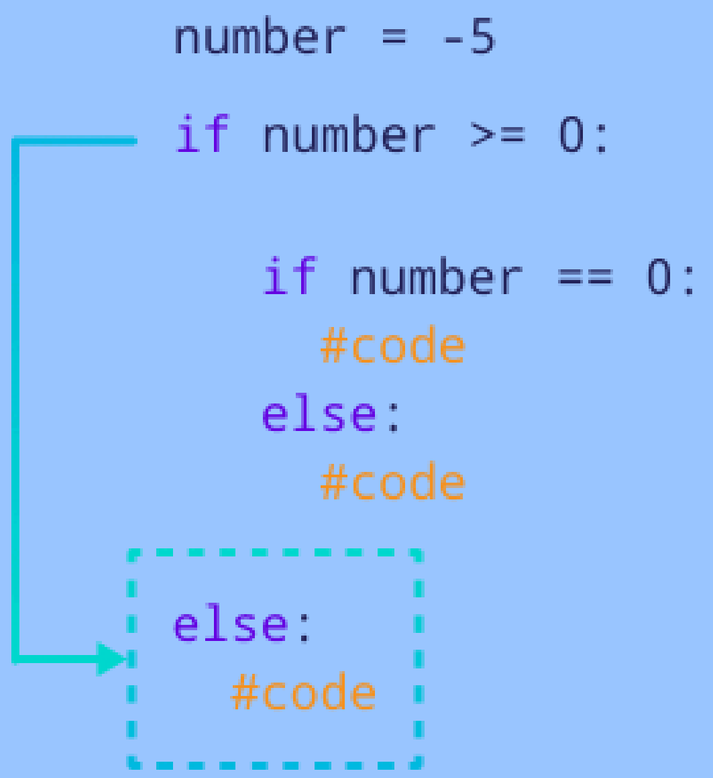
It is possible to include an if statement inside another if statement. For example,

```
number = 5

# outer if statement
if number >= 0:
    # inner if statement
    if number == 0:
      print('Number is 0')

    # inner else statement
    else:
        print('Number is positive')

# outer else statement
else:
    print('Number is negative')
```

Outer if Condition is True



Outer if Condition is False

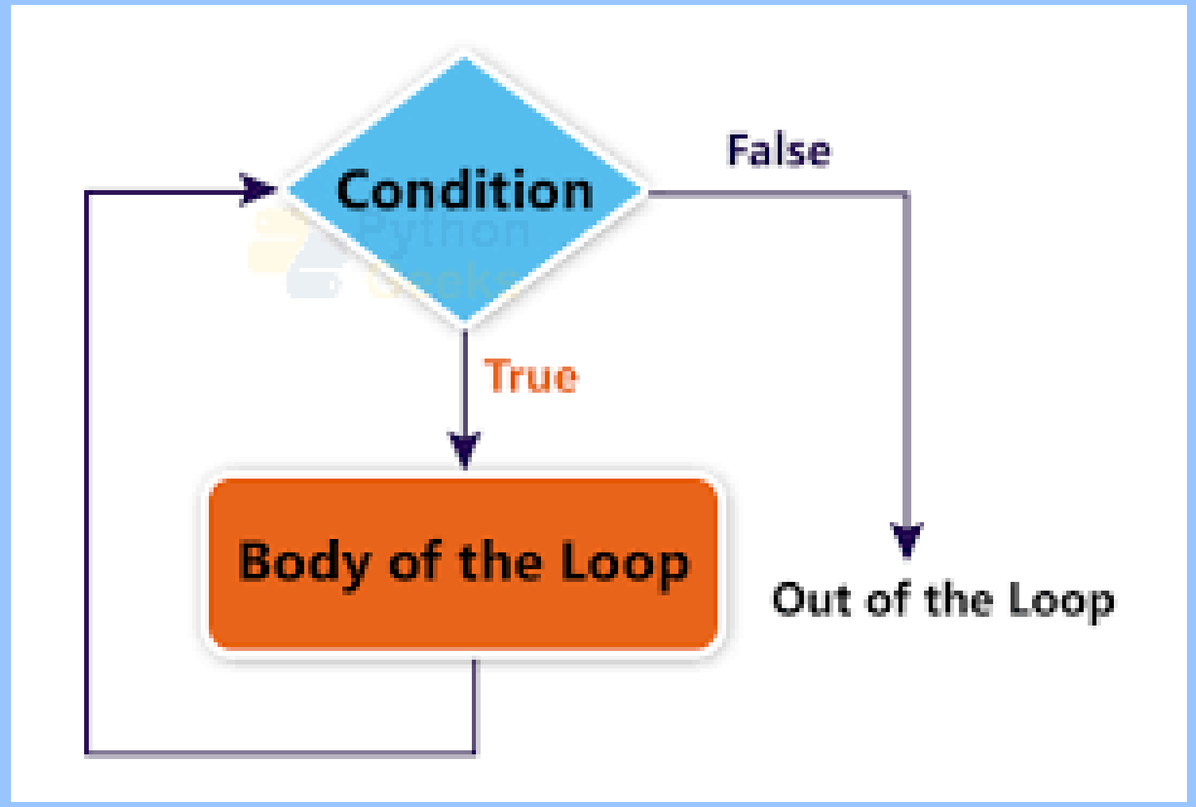


Loops

- Loops are used in programming to repeat a specific block of code.

Looping Constructs in Python

- **Whil e**
- **for**



While Loop

- The while loop in python is used to iterate over a block of code as long as the test expression (condition) is true.
- We generally use this loop when we don’t know the number of times to iterate

Syntax

```
while condition:
    # code to execute
```

Condition is a boolean expression that determines whether the loop should continue or not.

For example, let’s say we want to print the number from 1 to 5 using a while loop:

```
num = 1
while num <= 5:
    print(num)
    num += 1
```

In this example, we initialize the **num** variable to 1 and then execute the loop as long as **num** is less than or equal to 5. Inside the loop we print the current value of **num** and then increment it by 1

For loops

We use for loop when we want to iterate over a collection of items or when we know the exact number of times we want to execute a block of code

Here’s the code for a for loop in python

```
for variable in iterable:
    # code to execude
```

- **Variable** is a variable that represents the current item in the iterable that we’re iterating over
- **iterable** is a collection of items that we want to iterate over, such as list, tuple, string, or range

Loop continues until we reach the last item in the sequence.

```
# list of numbers
numbers = [5, 6, 7, 3, 2, 1]
# iterate over the list
for val in numbers:
    print(val)
```

Range Function:

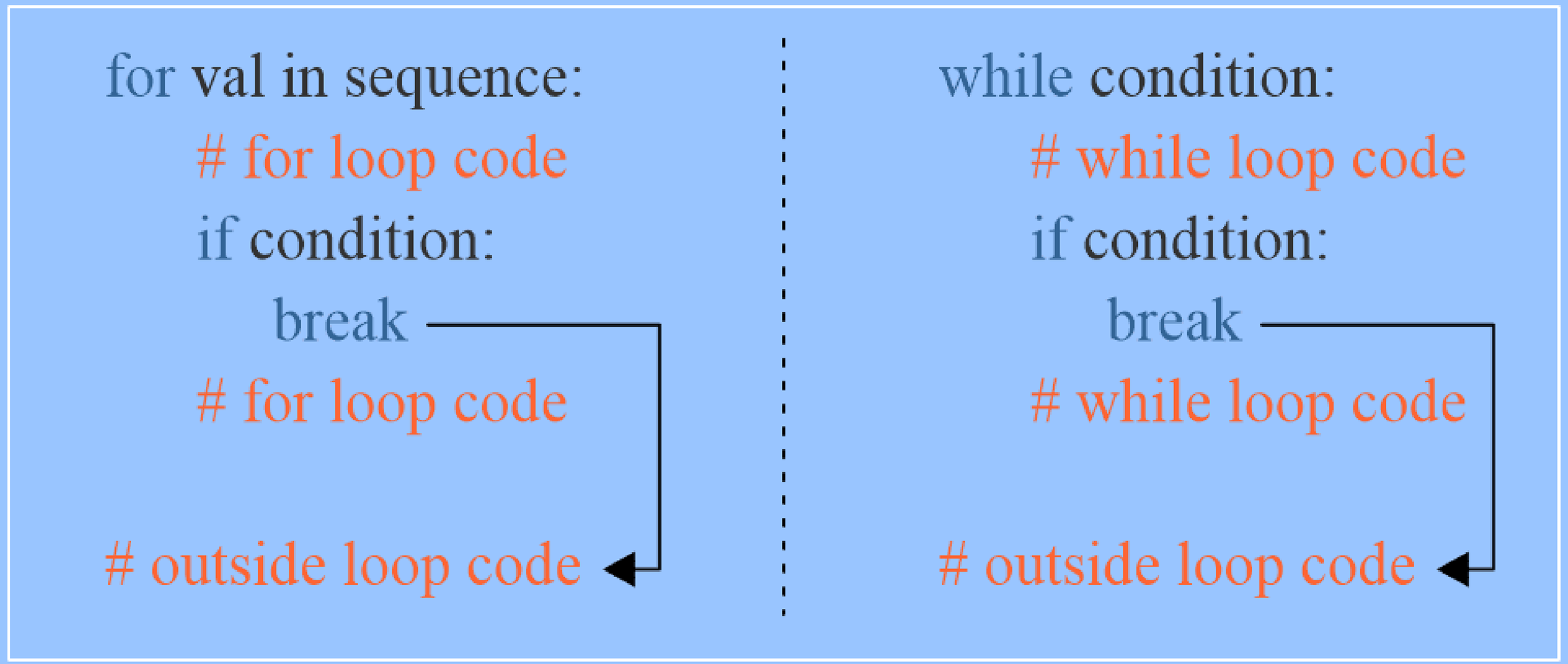
- We can generate a sequence of numbers using **range()** function.
- range(10) will generate numbers from 0 to 9 (10 numbers)

break and continue

- Loops iterate over a block of code until text expression is false, but sometimes we wish to terminate the current iteration or even the whole loop without checking test expression.

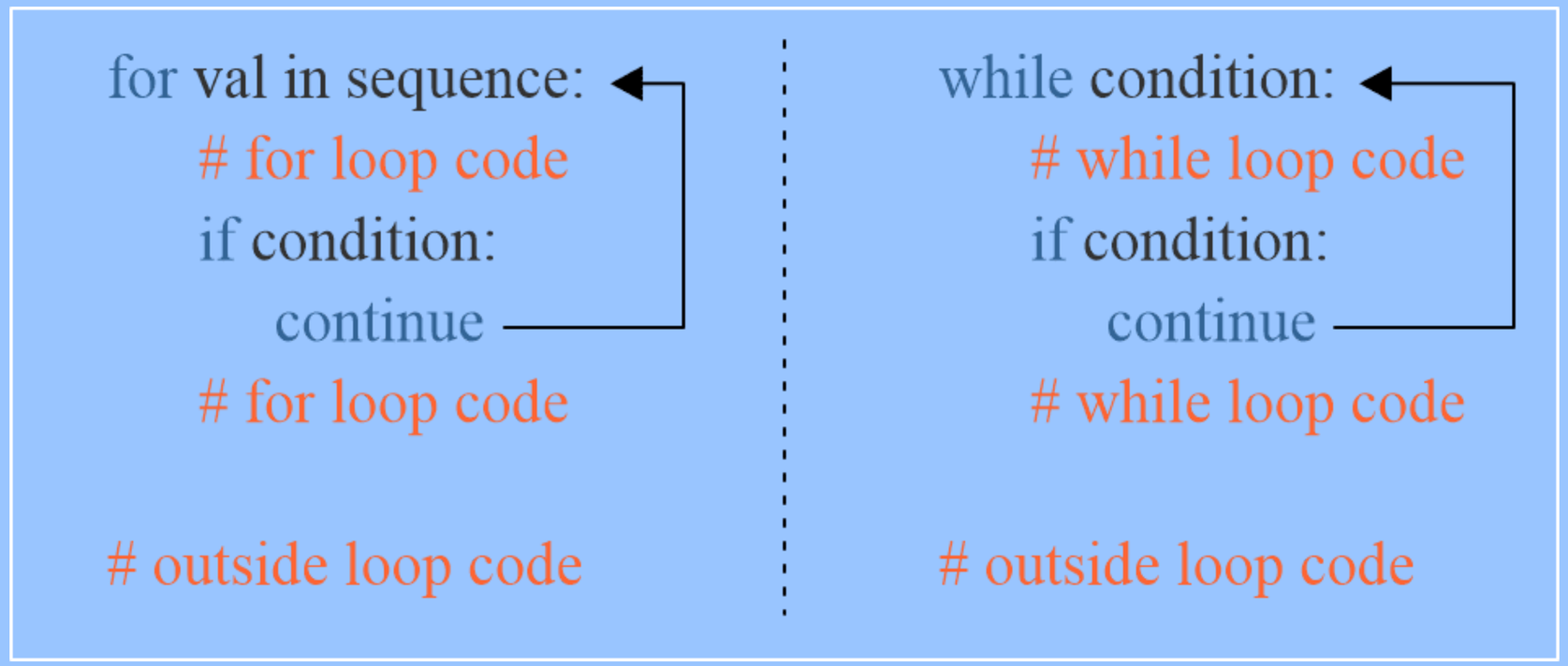
break statement:

- The break statement terminates the loop containing it. Control of the program flows to the statement immediately after the body of the loop.
- If break statement is inside a nested loop (loop inside another loop), break will terminate the innermost loop.


for val in sequence:
# for loop code
if condition:
break
# for loop code
# outside loop code
while condition:
# while loop code
if condition:
break
while loop code
outside loop code

Continue Statement:

- The continue statement is used to skip the rest of the code inside a loop for the current iteration only. Loop does not terminate but continues on with the next iteration.



Pass Statement

- Suppose we have a loop or a function that is not implemented yet, but we want to implement it in the future.
- They cannot have empty body.
- We use the pass statement to construct a body that does nothing.

```
# pass is just a placeholder for functionality to be added later.

sequence = {’p’,  ‘a’, ‘s’, ‘s’}
for val in sequence:
    pass
```

Let’s Practice

- WAP to check if a number entered by the user is odd or even.
- WAP to find the greatest of 3 numbers entered by the user.
- WAP to check if a number is a multiple of 7 or not.
- WAP to find the sum of first n numbers. (using while)
- WAP to find the factorial of first n numbers. (using for)
- Ask the user to enter a number and check whether it is positive, negative, or zero.
- Take a number as input and print its multiplication table up to 10. (Assignment)