Computational Thinking with Programming

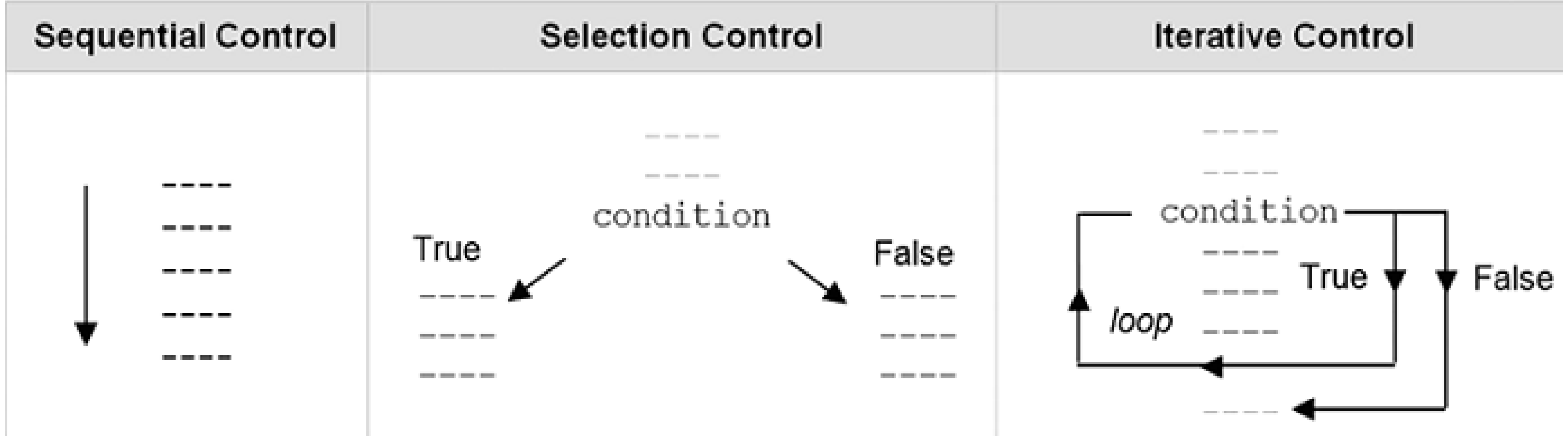
@cse_bennett @csebennett

Control Structures

Control Structures

- Control flow is the order that instructions are executed in a program.
- A control statement is a statement that determines the control flow of a set of instructions.
- Types of Control:
  - **Sequential control**: Instructions are executed in the order that they are written
  - **Selection control**: Selectively executes the instructions.

    E.g. Decision Control
  - **Iterative control**: Repeatedly executes the instructions. **E.g**. Loops.

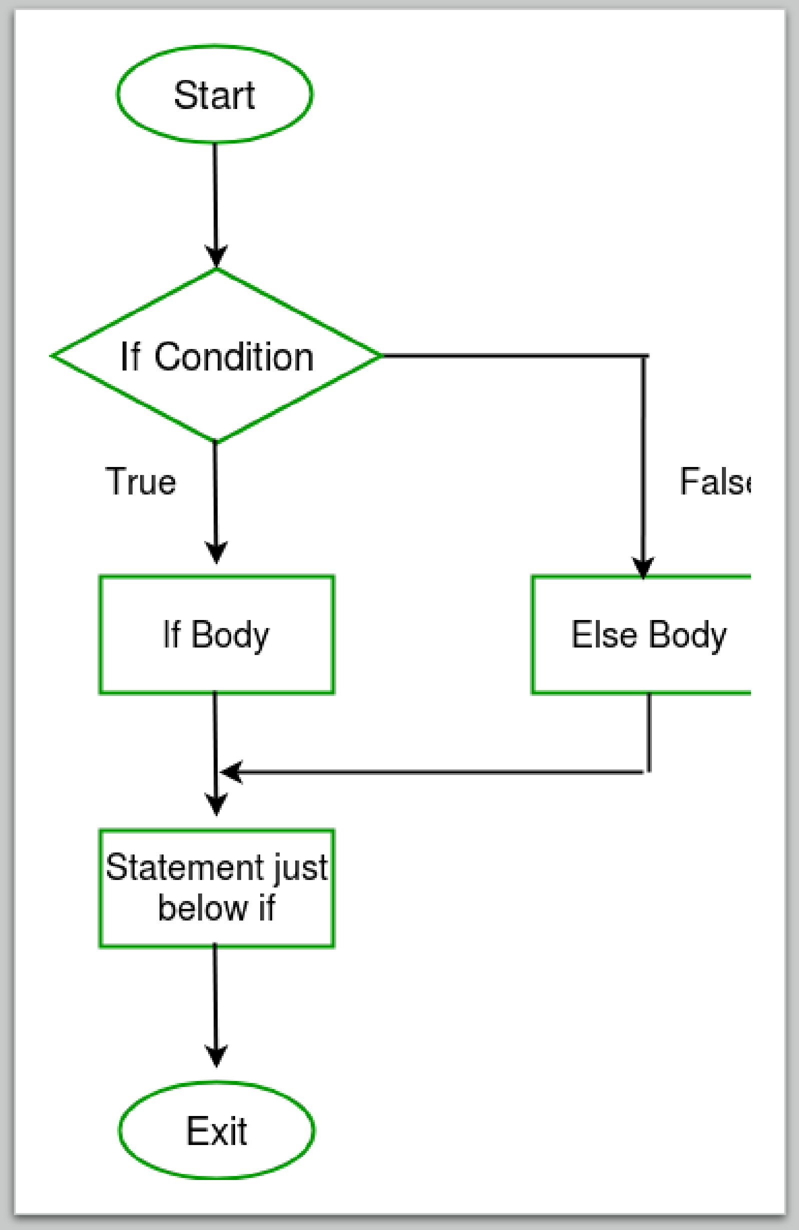


Selection Control or Decisions

(It is a control statement providing selective execution of instructions)

Decisions in a Python program

- if statements
- if else statements
- elif statements
- nested if conditions



If Statement

It is a selection control statement based on the value of a given Boolean expression

Syntax:

```
if test expression:
   statement(s)
```

Expression’s value can be True or False.
We may want to do something only when a certain condition is true.

if statement in python takes an expression with it.

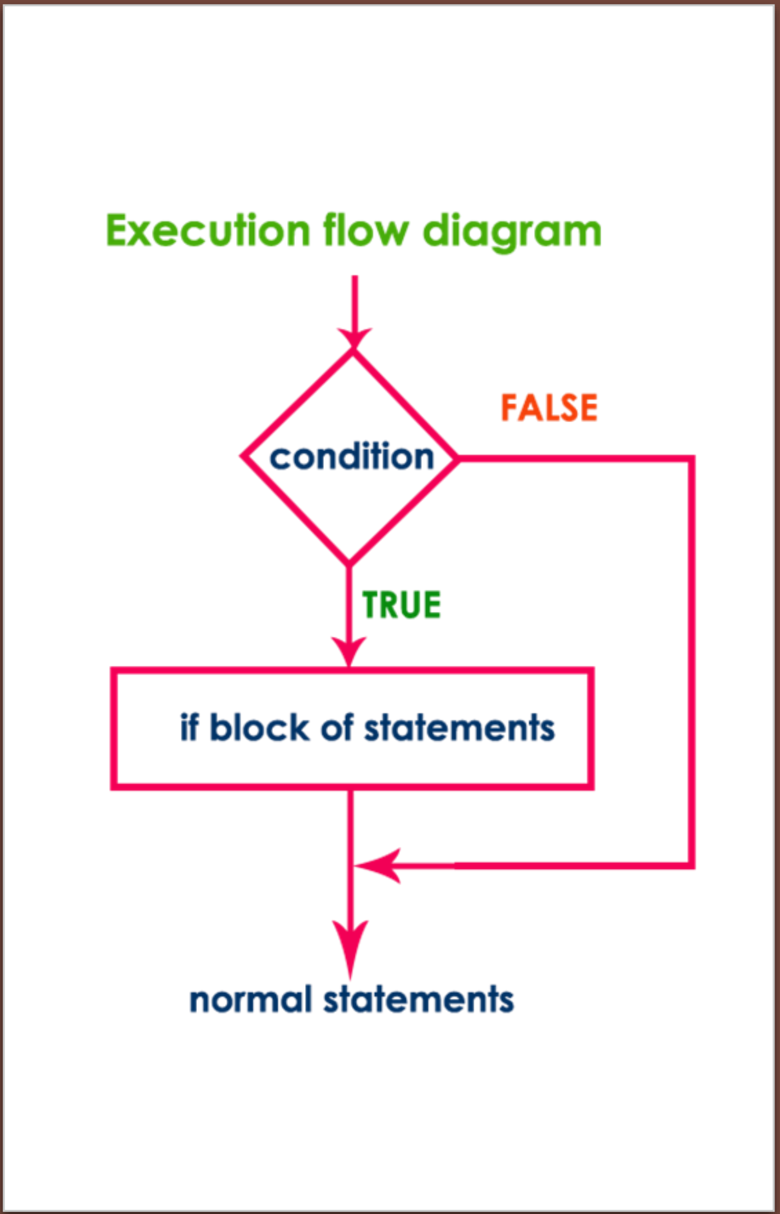
If the expression results to True

- then the block of statements under it is executed.

If it results to False

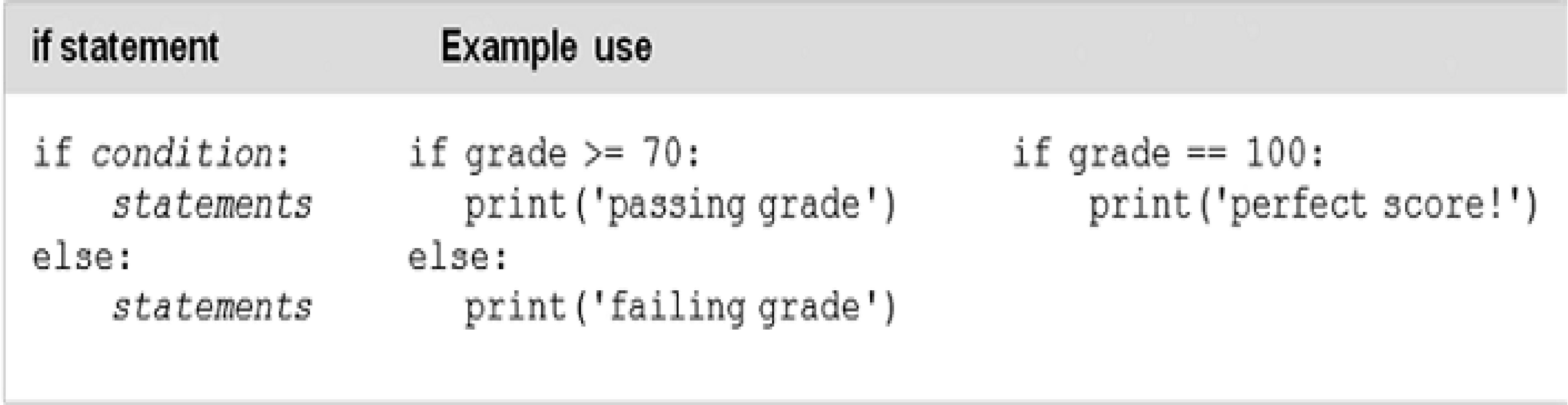
- then the block is skipped and control transfers to the statements after the block.

Remember to indent the statements in a block equally



Execution flow diagram

If Statement: Example

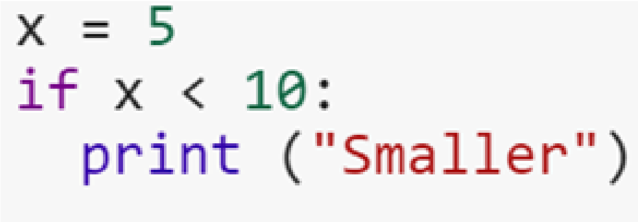


if statement	Example use	

if *condition*:
    statements
else:
    statements
	if grade >= 70:
	    print('passing grade')
	else:
	    print('failing grade')
		if grade == 100:
		    print('perfect score!')

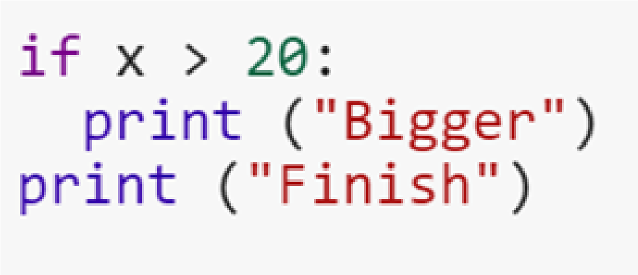
Example: What will be the output ?

1.



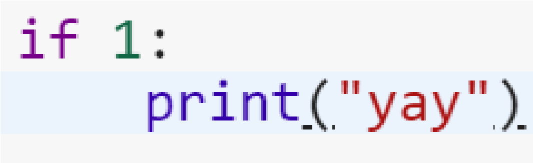
```
x = 5
if x < 10:
 print ("Smaller")
```

2.



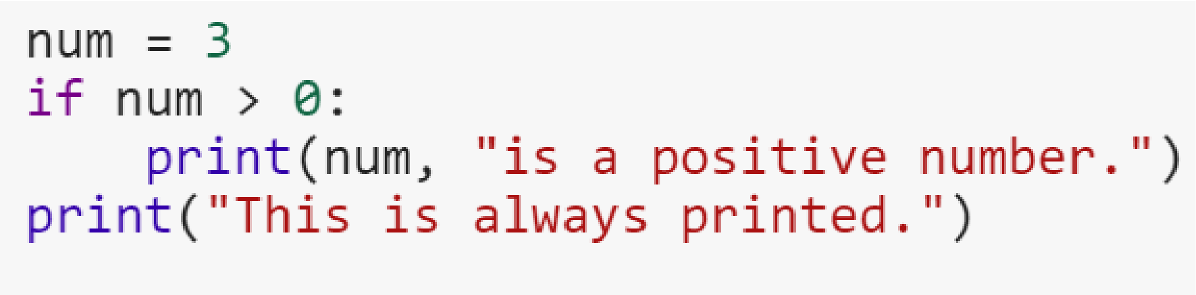
```
if x > 20:
 print ("Bigger")
print ("Finish")
```

3.



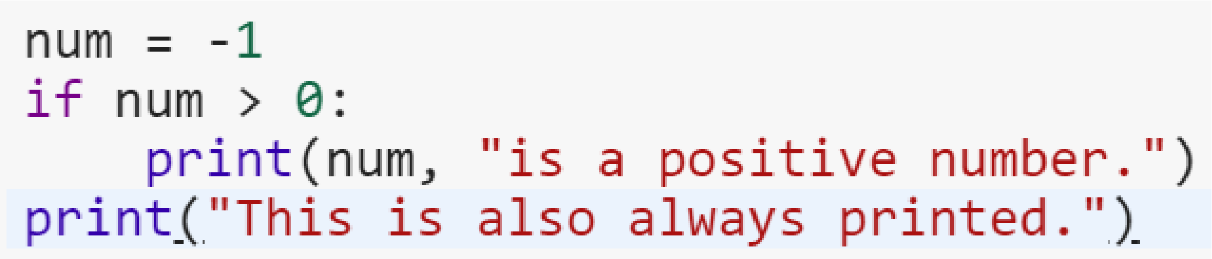
```
if 1:
    print("yay")
```

4.



```
num = 3
if num > 0:
    print(num, "is a positive number.")
print("This is always printed.")
```

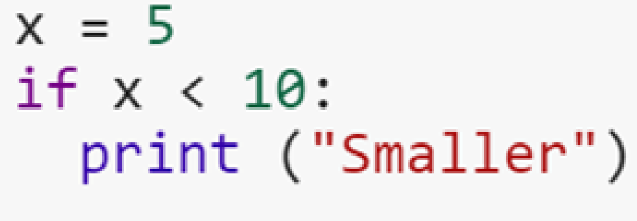
5.



```
num = -1
if num > 0:
    print(num, "is a positive number.")
print("This is also always printed.")
```

Example: What will be the output ?

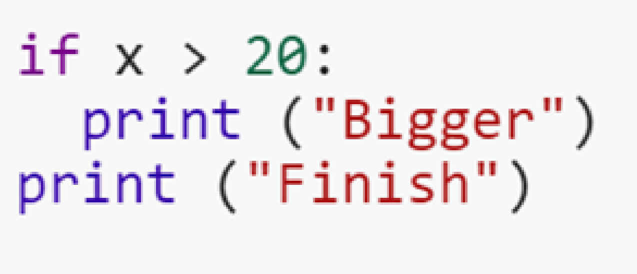
1.



```
x = 5
if x < 10:
  print ("Smaller")
```

Output: Smaller

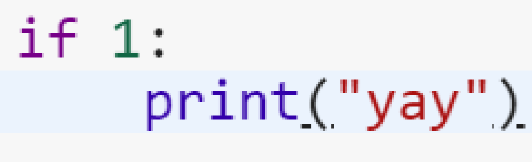
2.



```
if x > 20:
  print ("Bigger")
print ("Finish")
```

Output: Finish

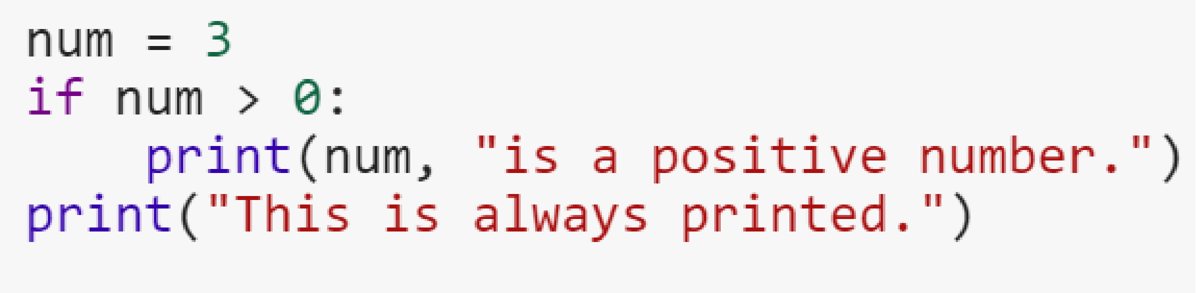
3.



```
if 1:
    print("yay")
```

Output: yay

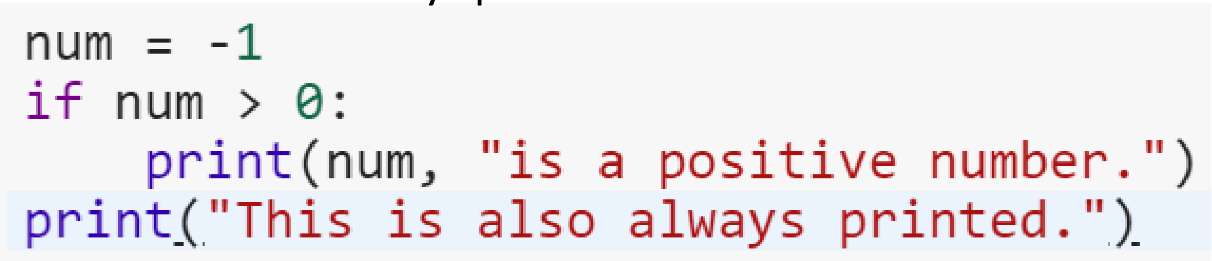
4.



```
num = 3
if num > 0:
    print(num, "is a positive number.")
print("This is always printed.")
```

Output: 3 is a positive number.
This is always printed.

5.



```
num = -1
if num > 0:
    print(num, "is a positive number.")
print("This is also always printed.")
```

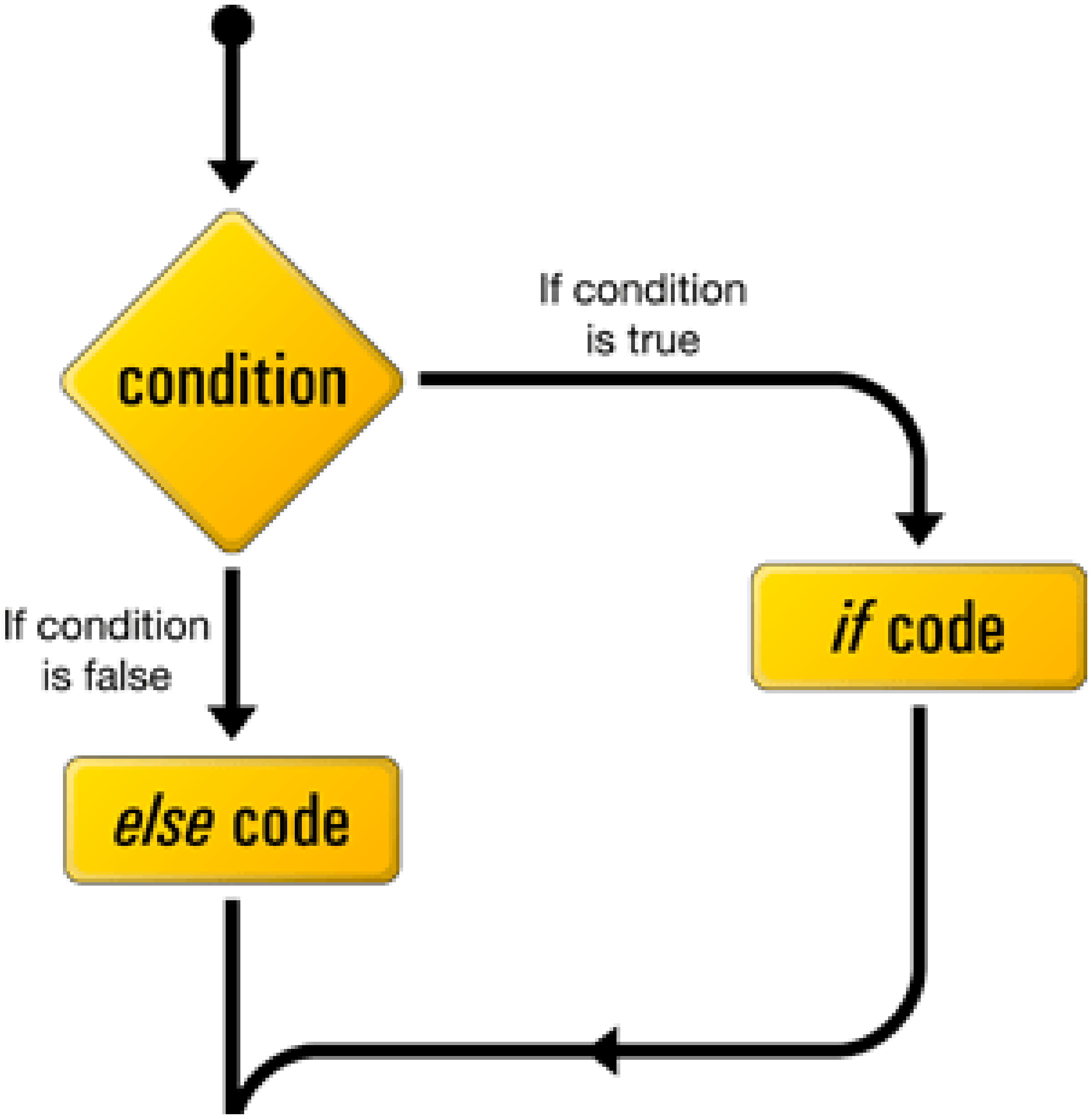
Output: This is also always printed.

if...else Statement
a.k.a. Two way decisions

What happens when the condition is untrue or false?

We can mention that it in the block after the else statement.

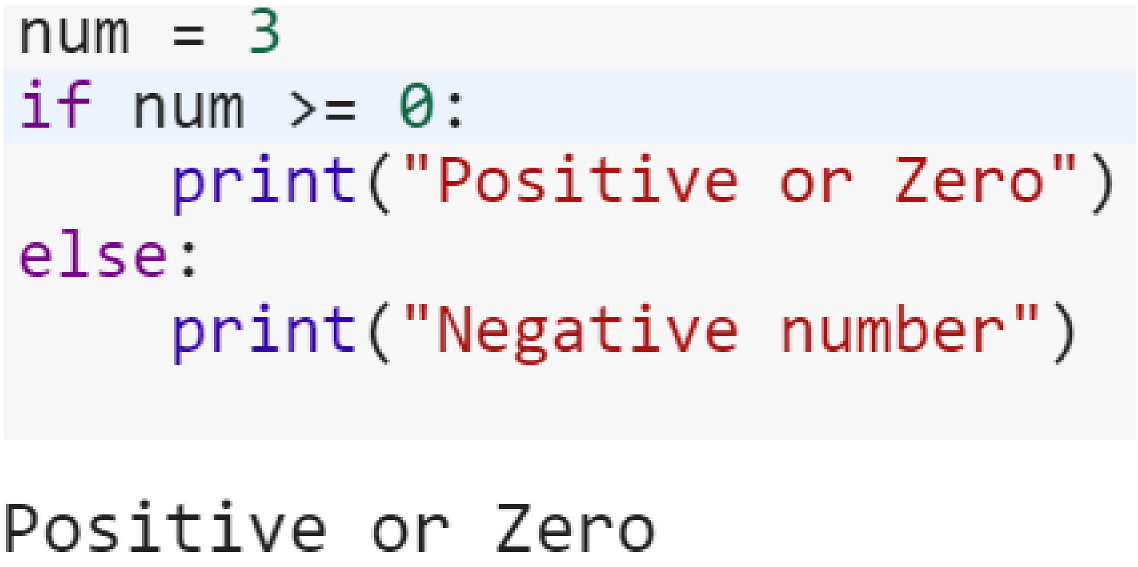
An ‘else’ statement comes right after the block after ‘if’.



Syntax:

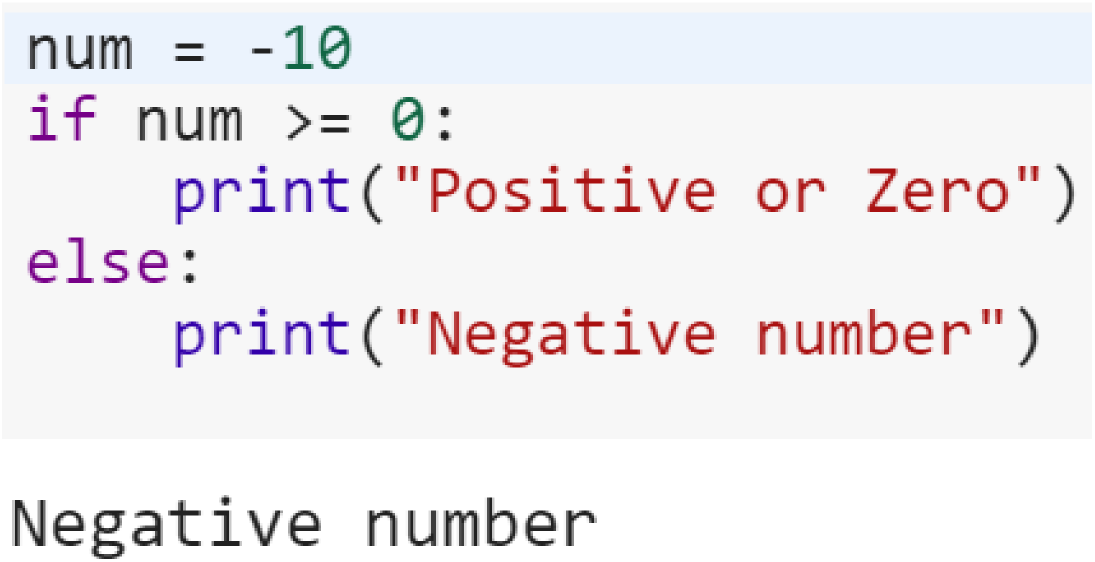
```
if test expression:
    Body of if
else:
    Body of else
```

Example



```python
num = 3
if num >= 0:
    print("Positive or Zero")
else:
    print("Negative number")
```

```
Positive or Zero
```

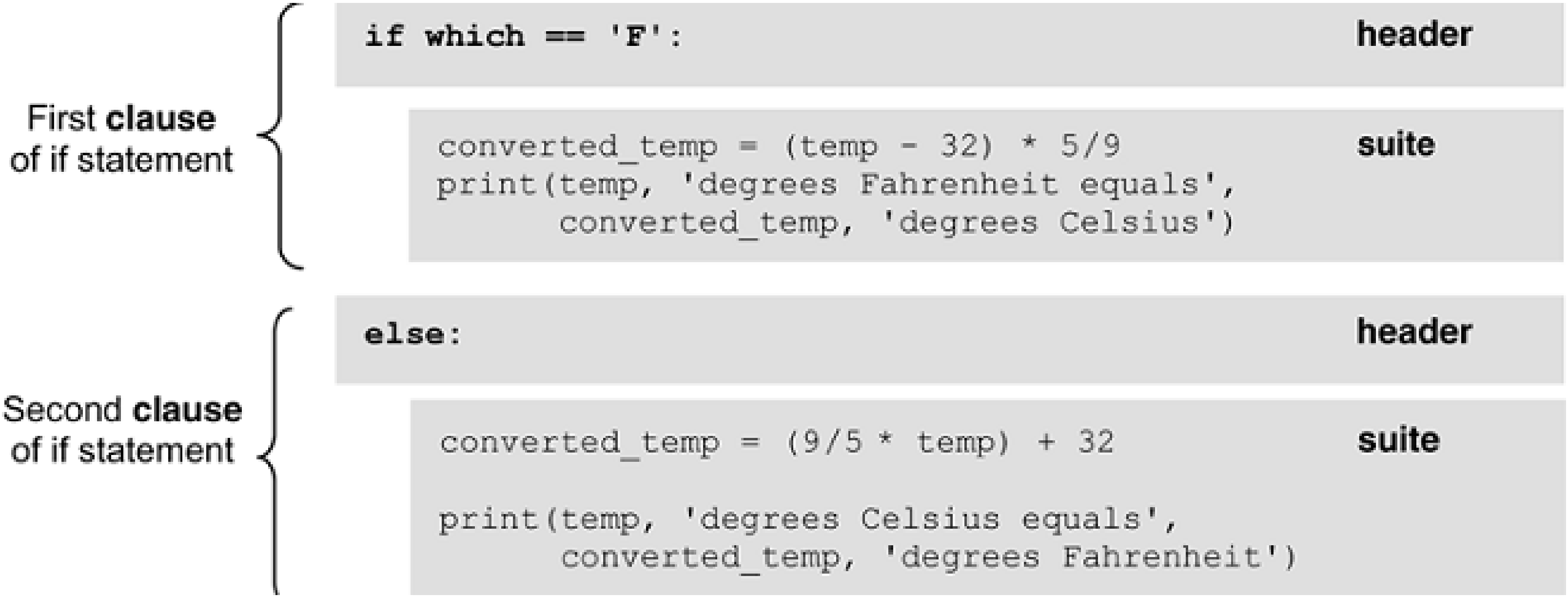


```python
num = -10
if num >= 0:
    print("Positive or Zero")
else:
    print("Negative number")
```

```
Negative number
```

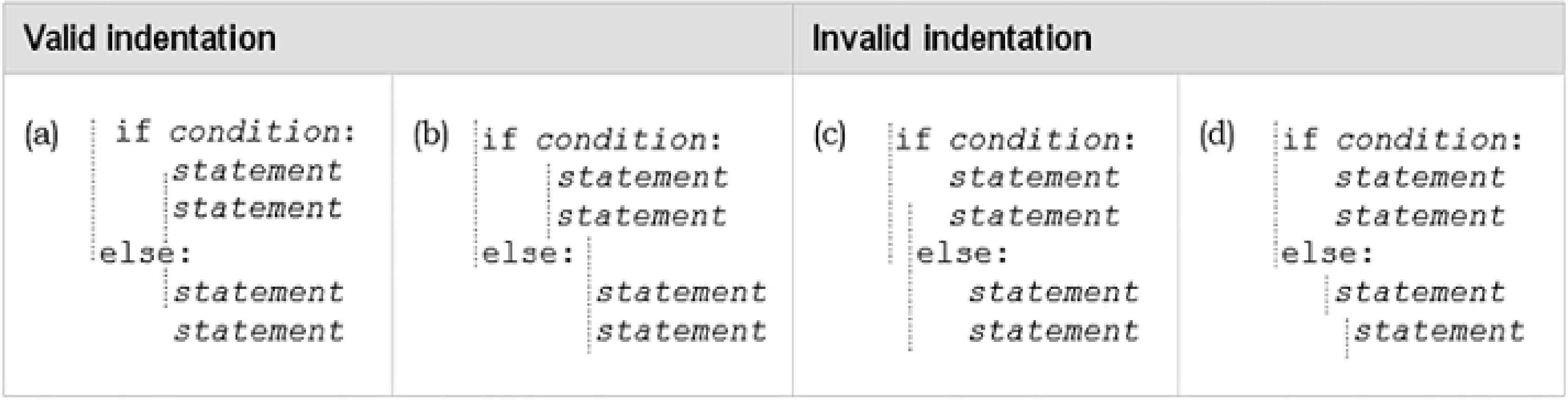
Header, Suite and Indentation

- One unique aspect of Python is that the amount of indentation of each program line is significant.



```
if which == 'F':
    converted_temp = (temp - 32) * 5/9
    print(temp, 'degrees Fahrenheit equals',
          converted_temp, 'degrees Celsius')
else:
    converted_temp = (9/5 * temp) + 32

    print(temp, 'degrees Celsius equals',
          converted_temp, 'degrees Fahrenheit')
```



Valid indentation

(a)

```
 if condition:
    statement
    statement
else:
    statement
    statement
```

(b)

```
if condition:
    statement
    statement
else:
      statement
      statement
```

Invalid indentation

(c)

```
if condition:
   statement
   statement
 else:
    statement
    statement
```

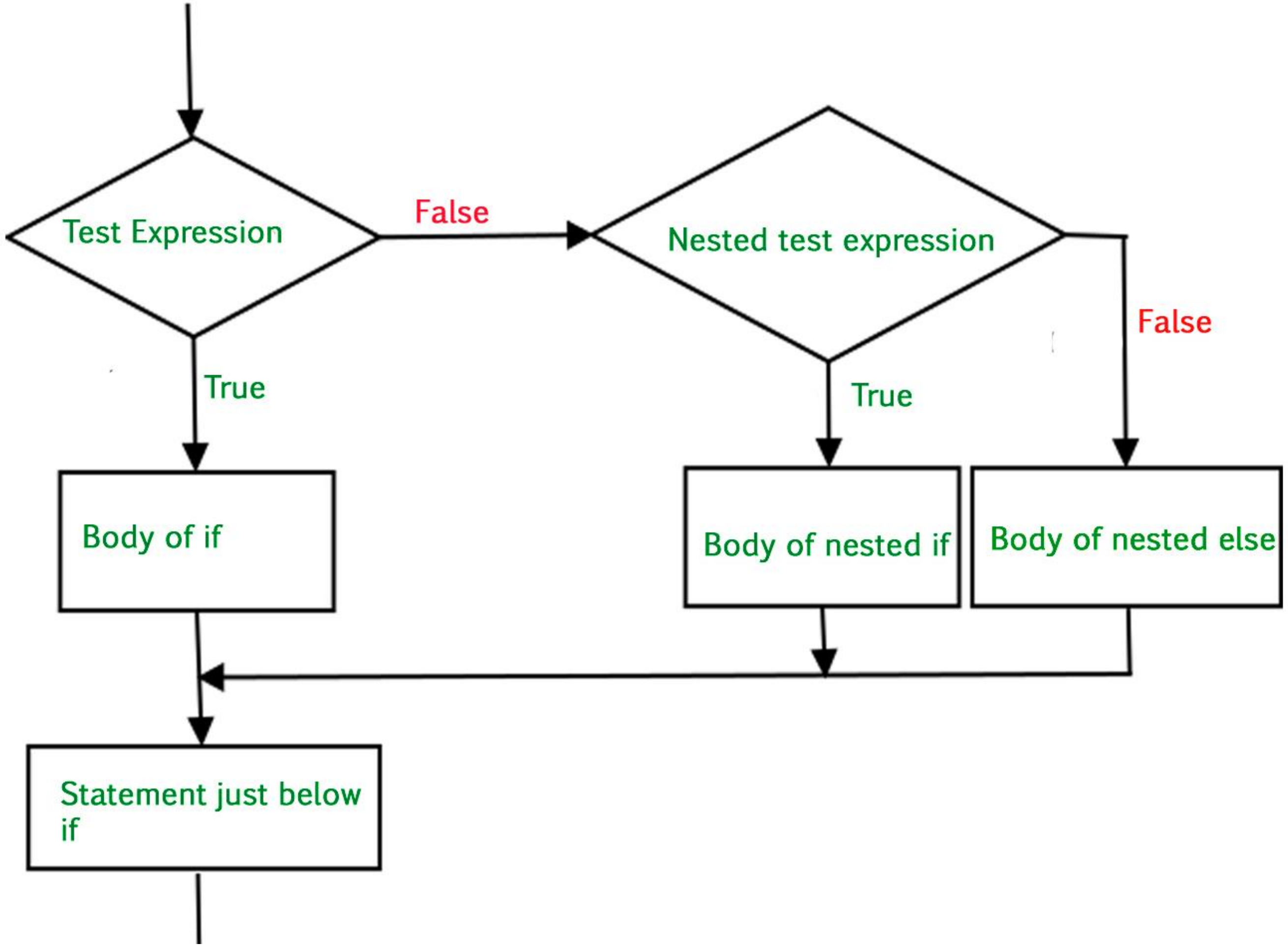
(d)

```
if condition:
   statement
   statement
else:
   statement
    statement
```

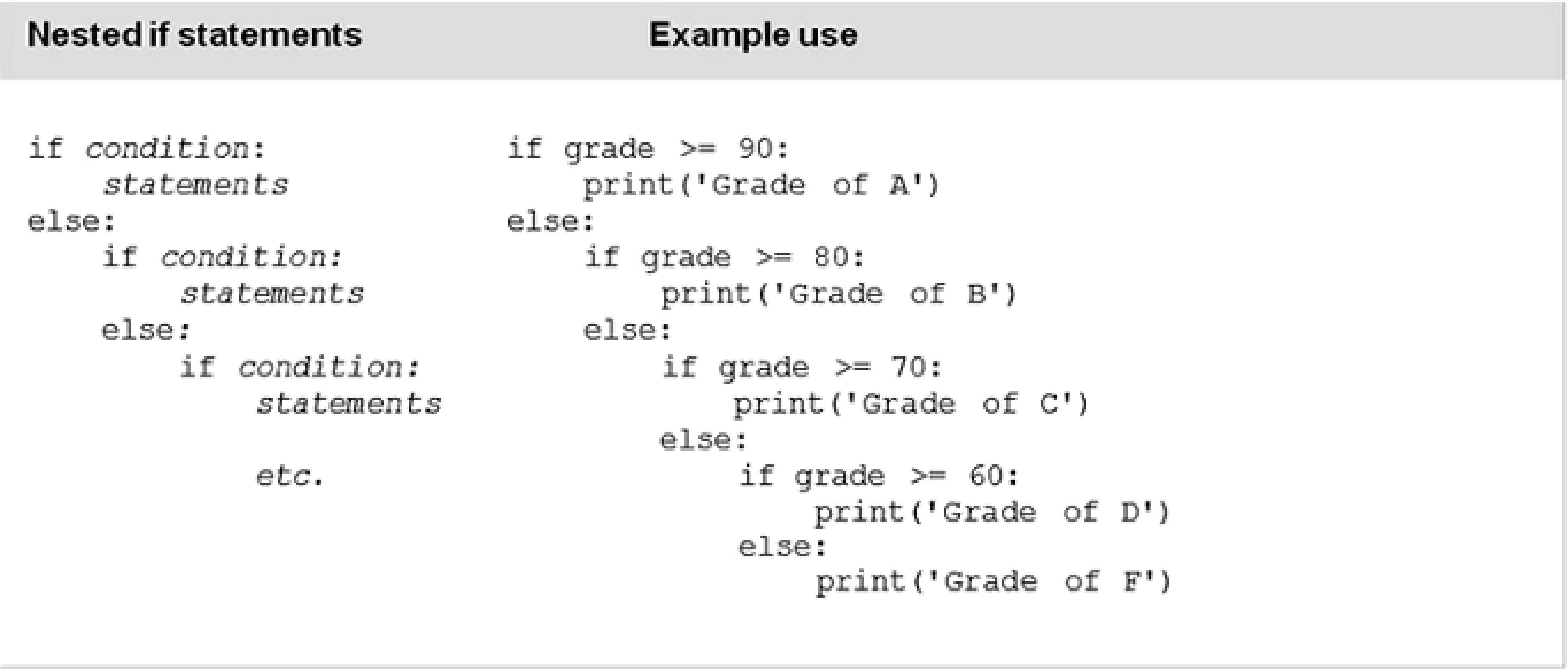
Nested if statements (multi-way selection)

You can put an if statement in the block under ***another if statement***.

This is to implement further checks.



Nested if statements: Example



Nested if statements

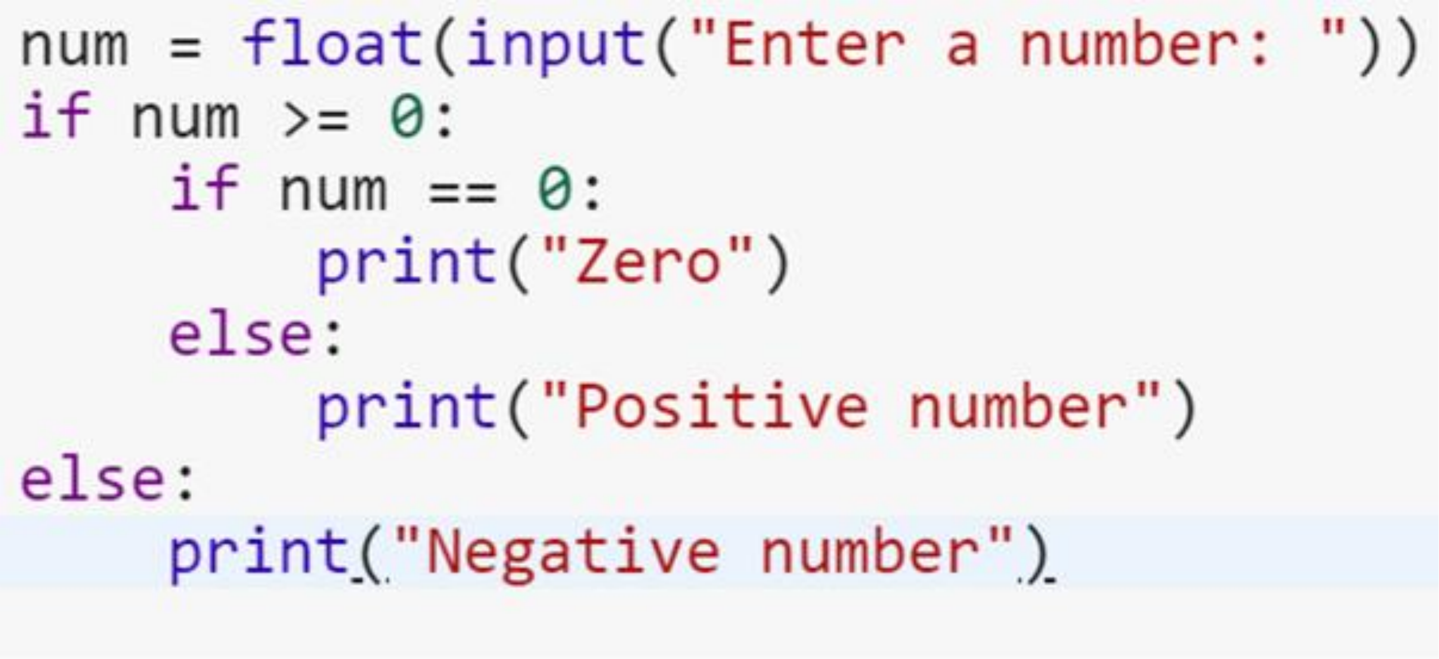
```
if condition:
    statements
else:
    if condition:
        statements
    else:
        if condition:
            statements

            etc.
```

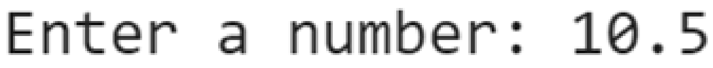
Example use

```
if grade >= 90:
    print('Grade of A')
else:
    if grade >= 80:
        print('Grade of B')
    else:
        if grade >= 70:
            print('Grade of C')
        else:
            if grade >= 60:
                print('Grade of D')
            else:
                print('Grade of F')
```

Example: What will be the output?

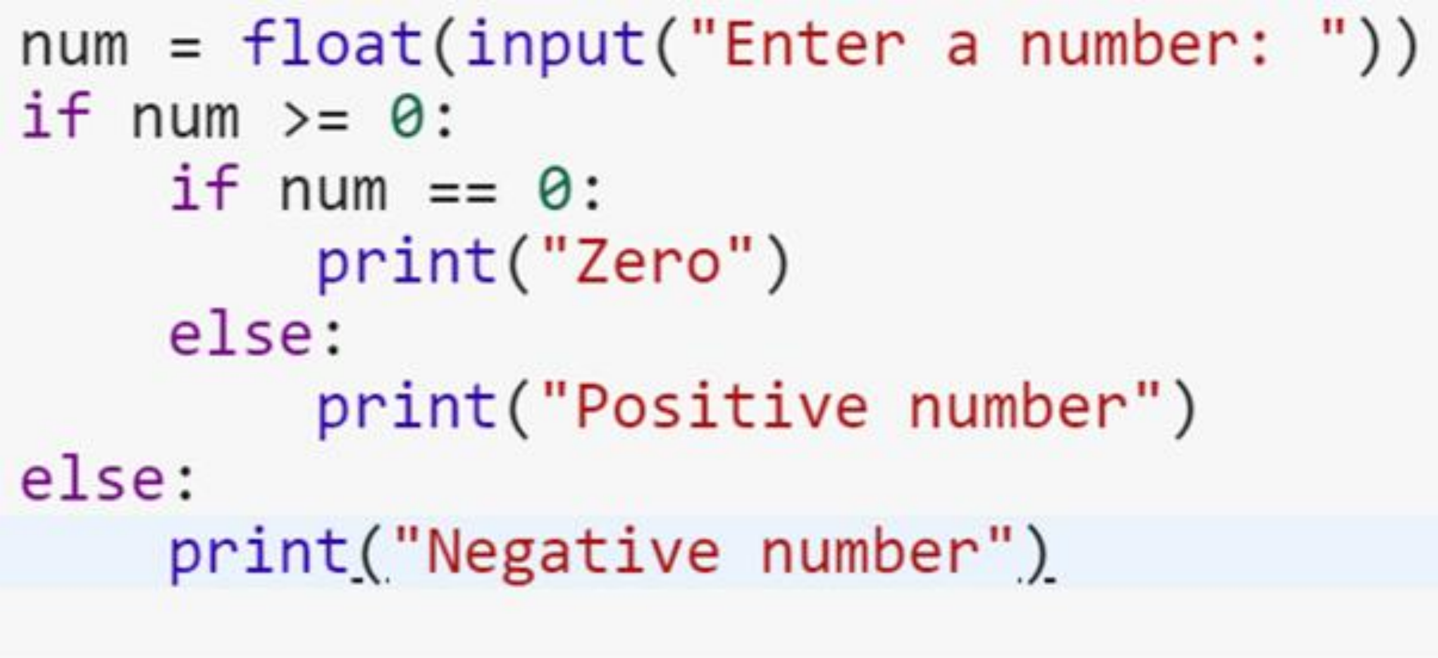


```python
num = float(input("Enter a number: "))
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```

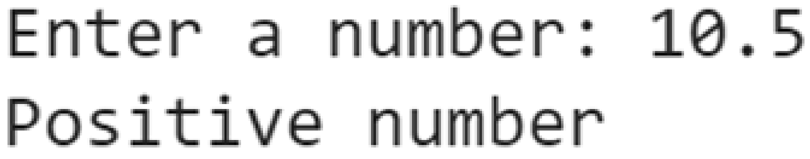


```
Enter a number: 10.5
```

Example: What will be the output?



```
num = float(input("Enter a number: "))
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```



```
Enter a number: 10.5
Positive number
```

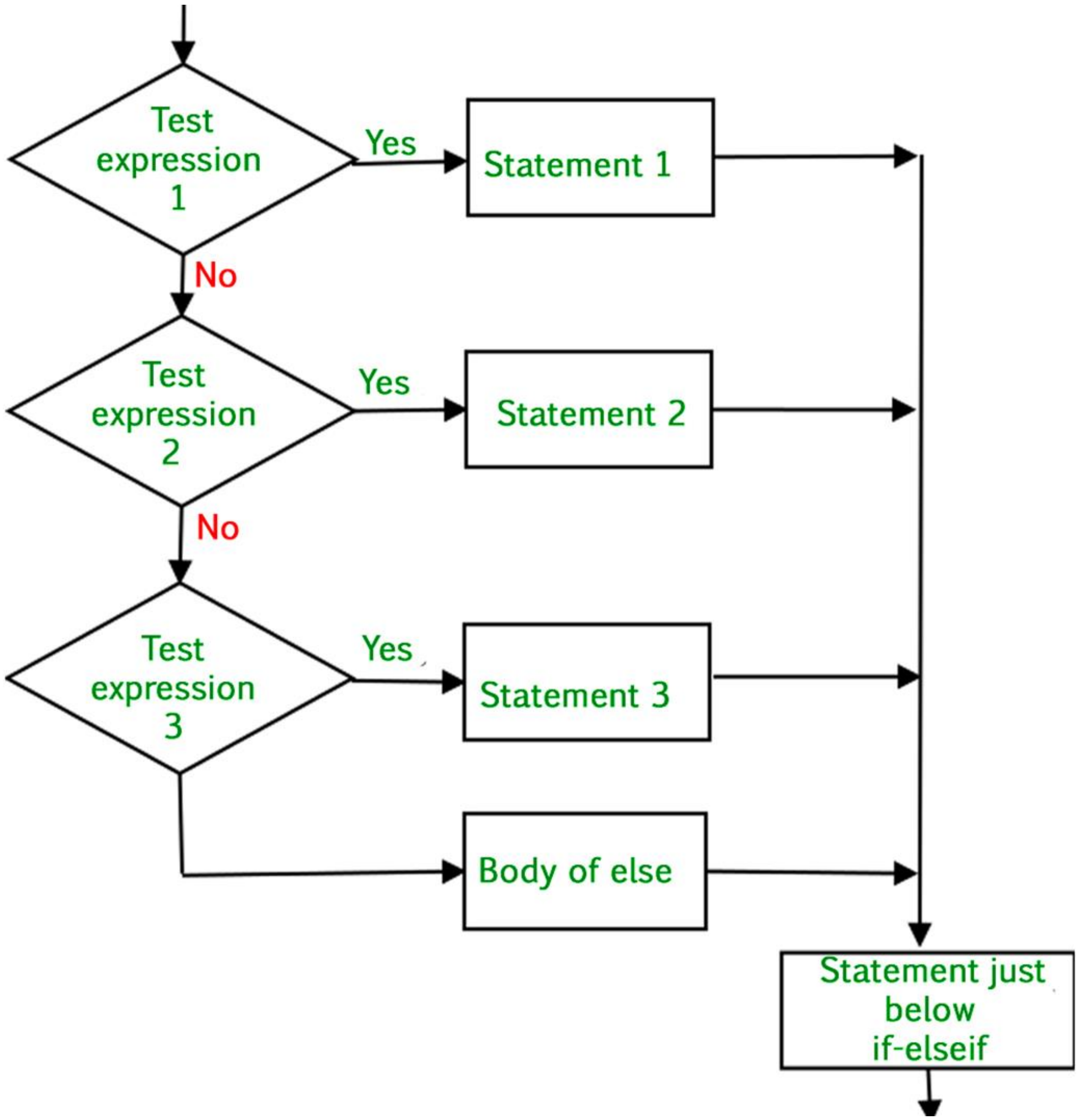
if...elif...else Statement

Replacement to the else-if statements

More than one condition to check

If condition 1 isn't True, ***condition 2*** is checked.

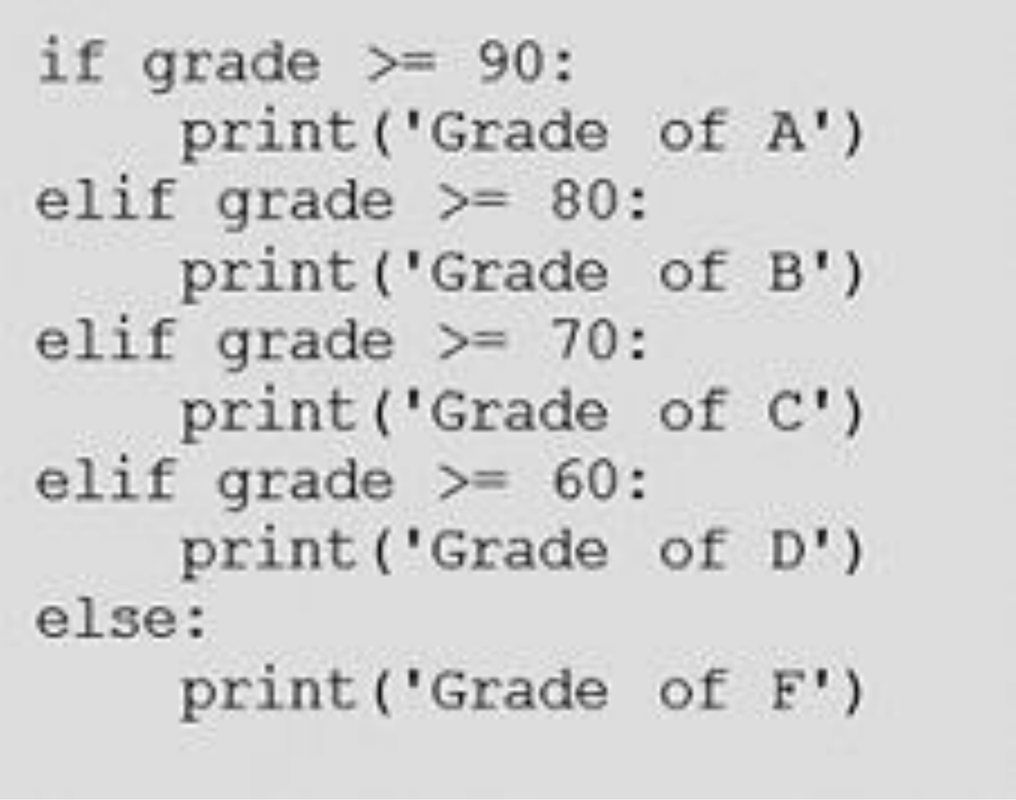
If it isn't true, ***condition 3*** is checked.



if...elif...else Statement: Example Use

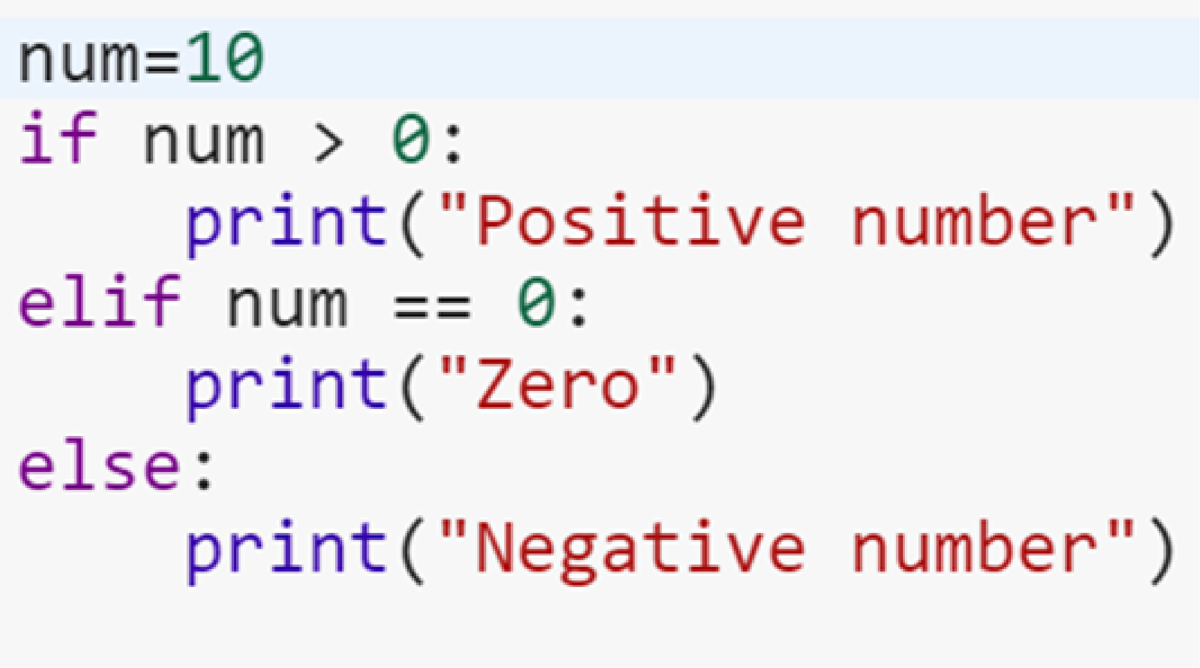
Syntax:

```
if test expression:
    Body of if
elif test expression:
    Body of elif
else:
    Body of else
```



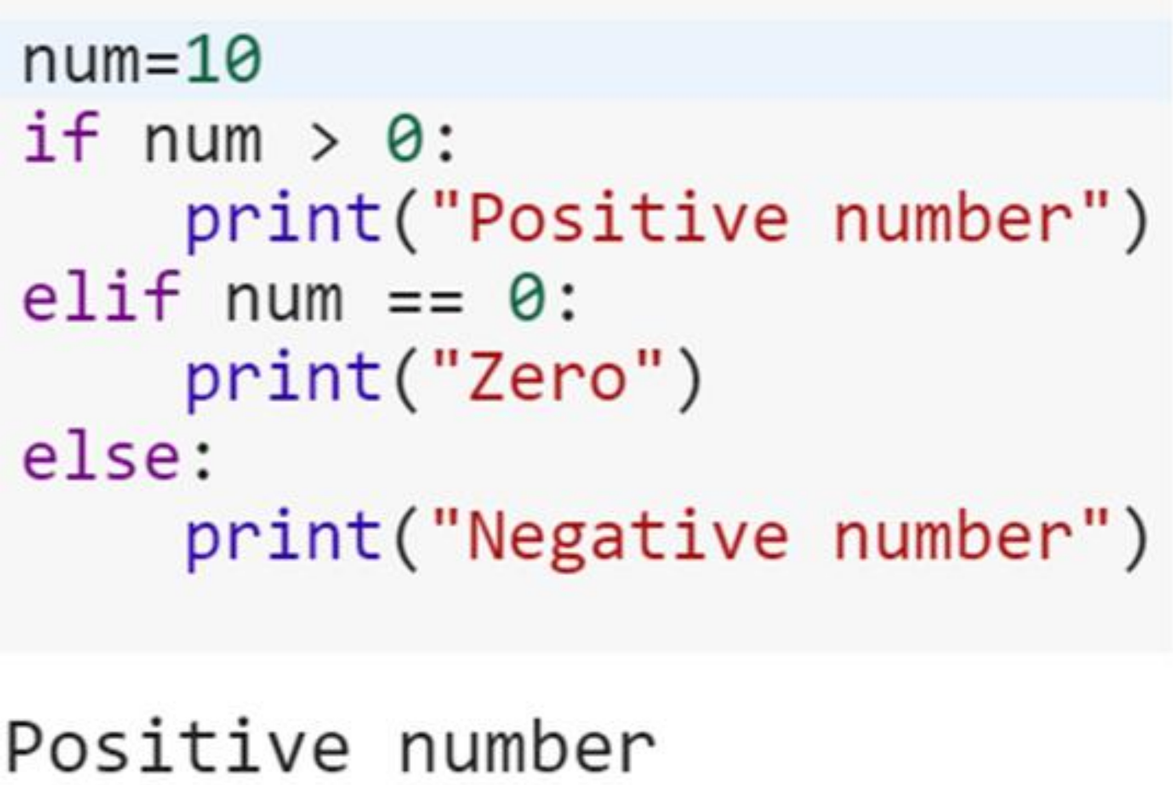
```
if grade >= 90:
    print('Grade of A')
elif grade >= 80:
    print('Grade of B')
elif grade >= 70:
    print('Grade of C')
elif grade >= 60:
    print('Grade of D')
else:
    print('Grade of F')
```

Example: What will be the output?



```python
num=10
if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
```

Example: What will be the output?

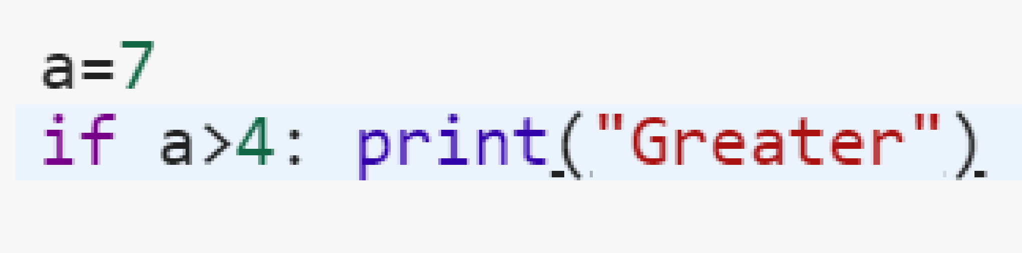


```
num=10
if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
```

```
Positive number
```

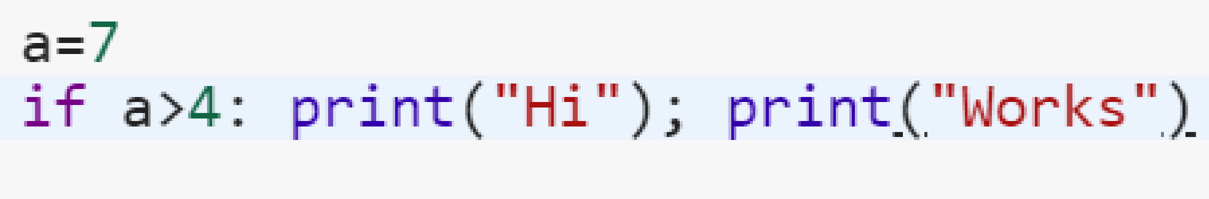
Single Statement Condition

Write a single statement under if



```
a=7
if a>4: print("Greater")
```

```
Greater
```



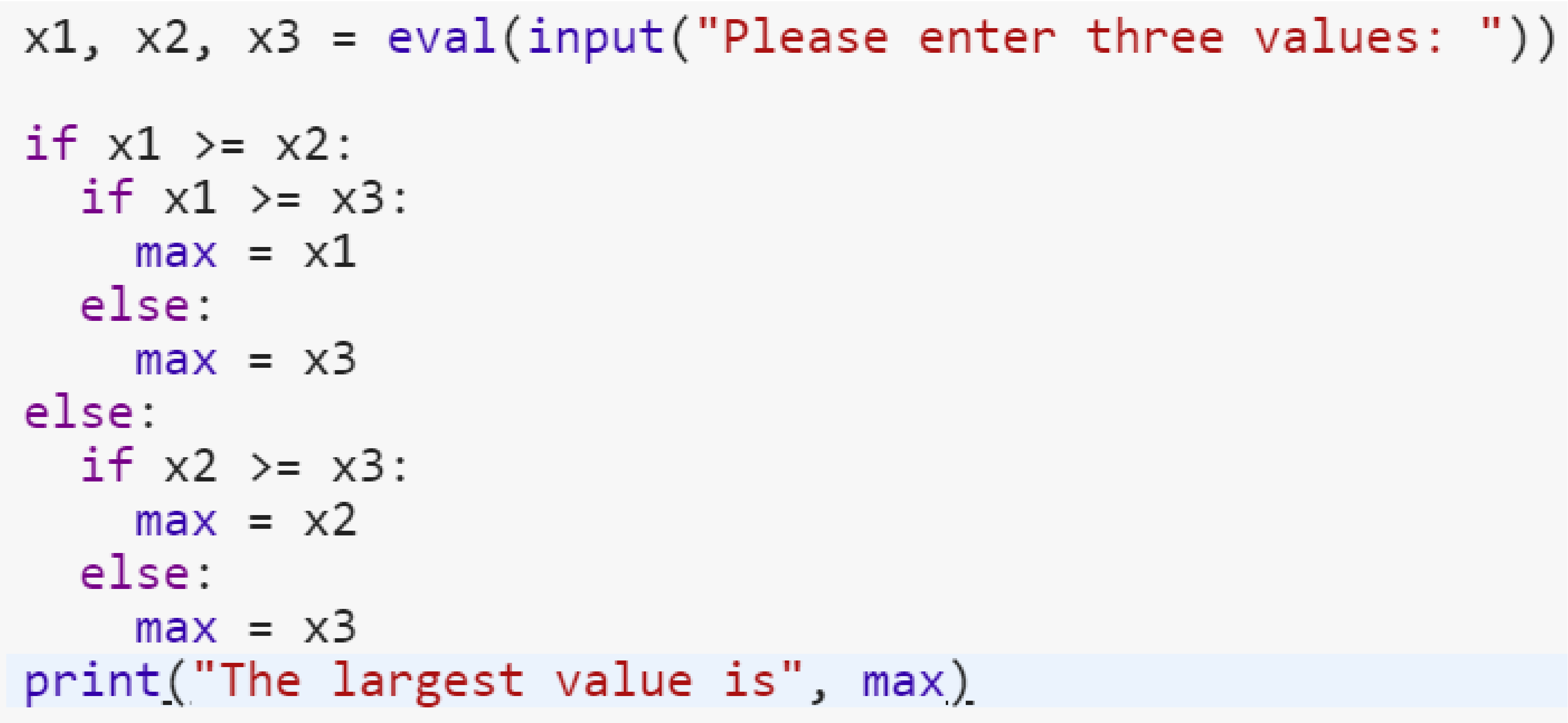
```
a=7
if a>4: print("Hi"); print("Works")
```

```
Hi
Works
```

Exercise

Find out the Max of three numbers

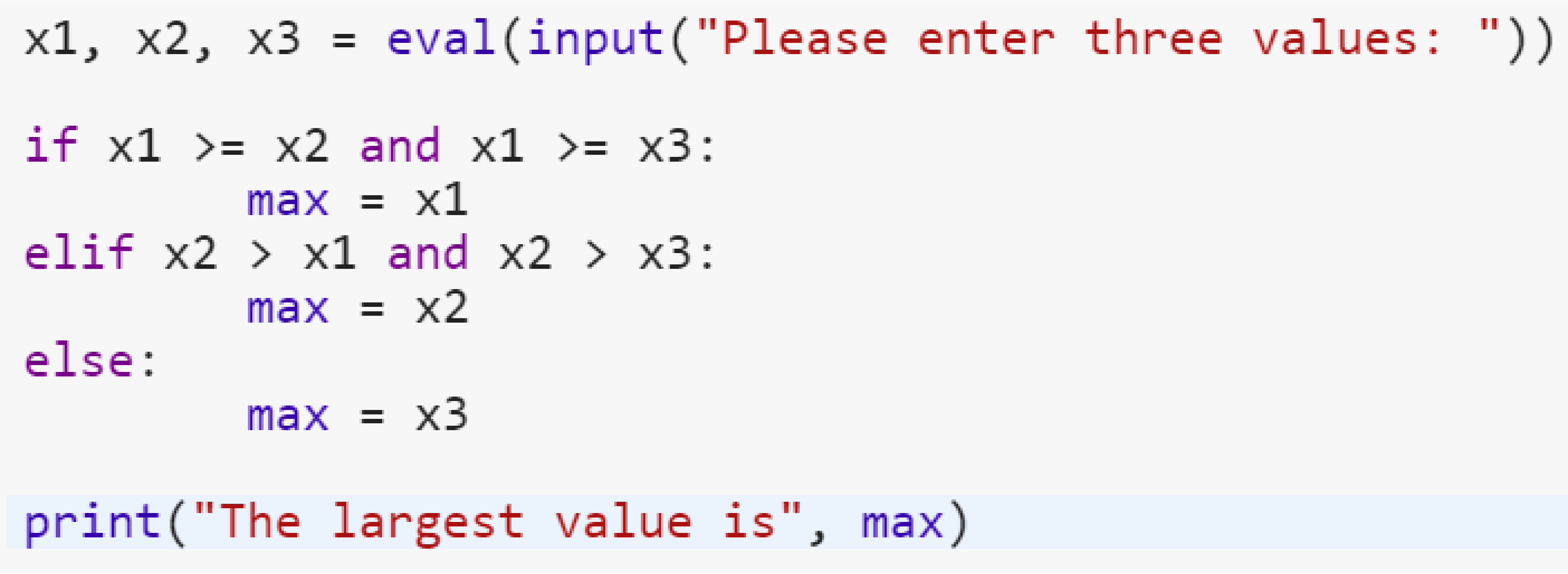
Maximum between 3 numbers



```
x1, x2, x3 = eval(input("Please enter three values: "))

if x1 >= x2:
  if x1 >= x3:
    max = x1
  else:
    max = x3
else:
  if x2 >= x3:
    max = x2
  else:
    max = x3
print("The largest value is", max)
```

Maximum between 3 numbers

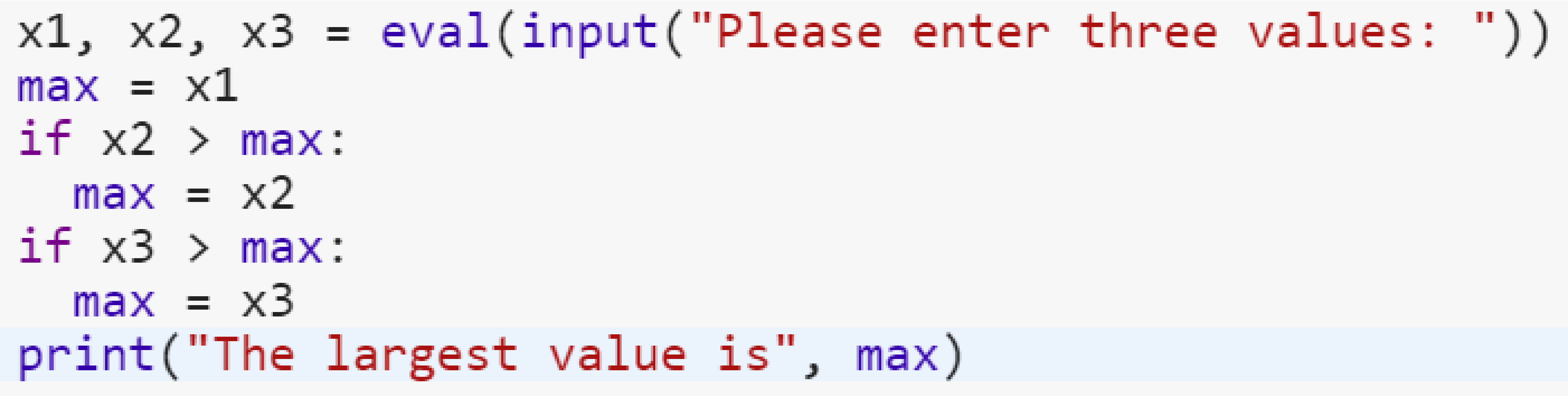


```python
x1, x2, x3 = eval(input("Please enter three values: "))

if x1 >= x2 and x1 >= x3:
        max = x1
elif x2 > x1 and x2 > x3:
        max = x2
else:
        max = x3

print("The largest value is", max)
```

Maximum between 3 numbers



```python
x1, x2, x3 = eval(input("Please enter three values: "))
max = x1
if x2 > max:
  max = x2
if x3 > max:
  max = x3
print("The largest value is", max)
```

MCQs

1. All if statements must contain either an else or elif header.
   - a) TRUE
   - [✓ b) FALSE]

2. Which of the following statements are true regarding headers in Python?
   - a) Headers begin with a keyword and end with a colon.
   - b) Headers always occur in pairs.
   - c) All headers of the same compound statement must be indented the same amount.

3. Which of the following statements is true?
   - a) Statements within a suite can be indented a different amount.
   - b) Statements within a suite can be indented a different amount as long as all headers in the statement that it occurs in are indented the same amount.
   - c) All headers must be indented the same amount as all other headers in the same statement, and all statements in a given suite must be indented the same amount.

4. The elif header allows for,
   - a) Multi-way selection that cannot be accomplished otherwise
   - b) Multi-way selection as a single if statement
   - c) The use of a “catch-all” case in multi-way selection

MCQs: Answers

1. All if statements must contain either an else or elif header.
   a) TRUE
   b) FALSE

2. Which of the following statements are true regarding headers in Python?
   a) Headers begin with a keyword and end with a colon.
   b) Headers always occur in pairs.
   c) All headers of the same compound statement must be indented the same amount.

3. Which of the following statements is true?
   a) Statements within a suite can be indented a different amount.
   b) Statements within a suite can be indented a different amount as long as all headers in the statement that it occurs in are indented the same amount.
   c) All headers must be indented the same amount as all other headers in the same statement, and all statements in a given suite must be indented the same amount.

4. The elif header allows for,
   a) Multi-way selection that cannot be accomplished otherwise
   b) Multi-way selection as a single if statement
   c) The use of a “catch-all” case in multi-way selection

Iterative Control (Loop)

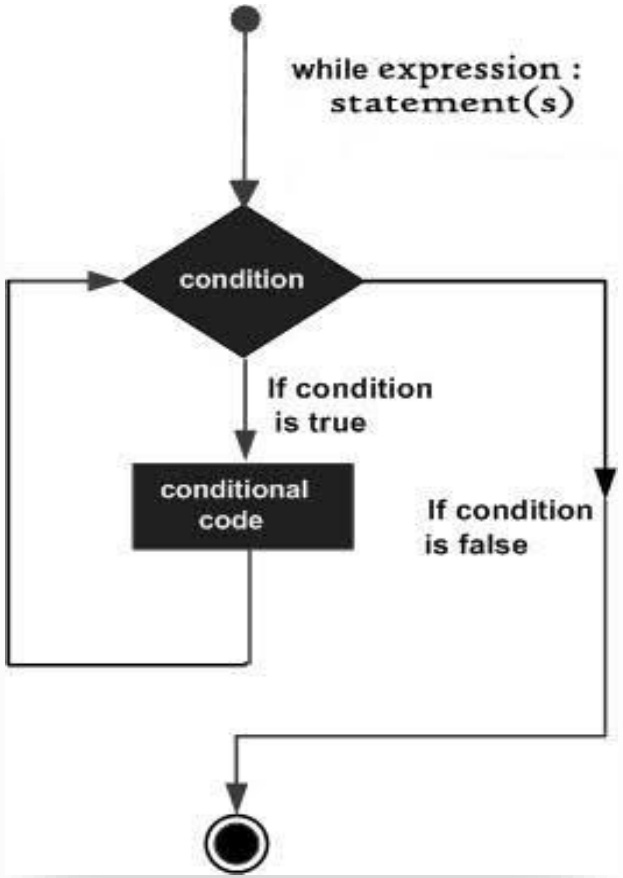
- An iterative control statement is a control statement that allows for the repeated execution of a set of statements.

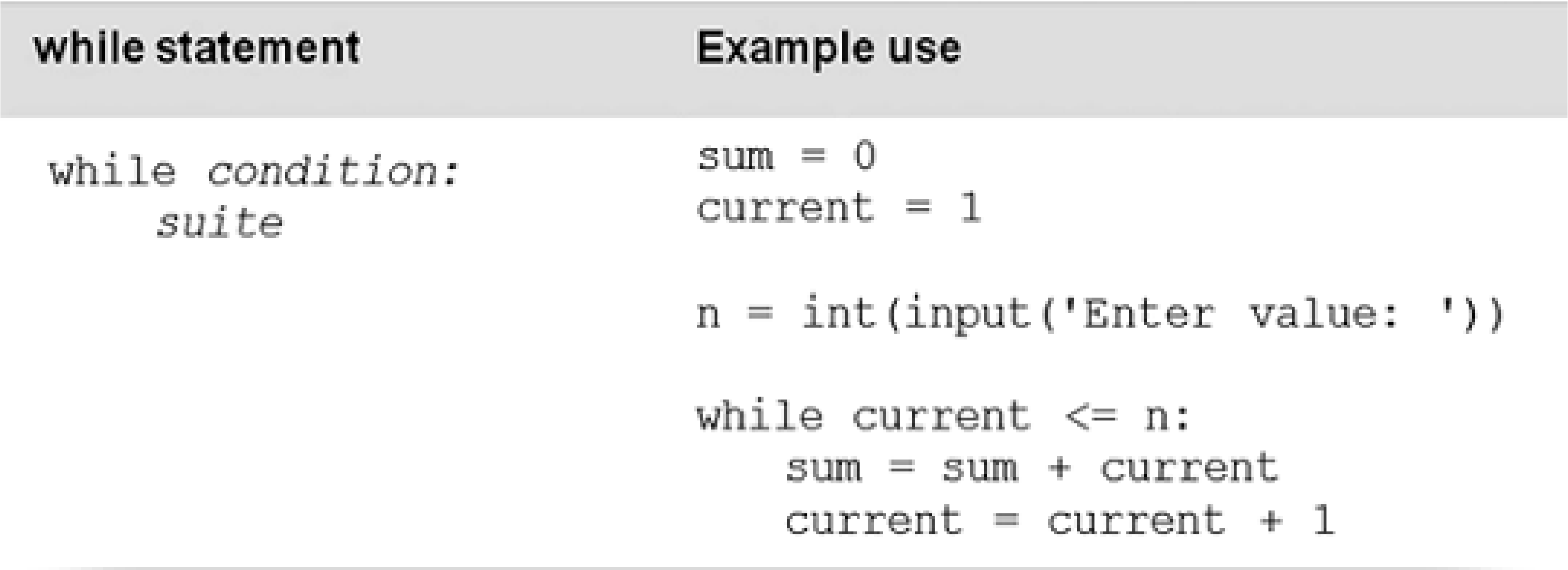
- Due to their repeated execution, iterative control structures are commonly referred to as “loops.

- Loop Statements:
  - While
  - For
  - Nested loop

While Statement (indefinite loop)

- A **while statement** is an iterative control statement that repeatedly executes a set of statements based on a provided Boolean expression (condition).



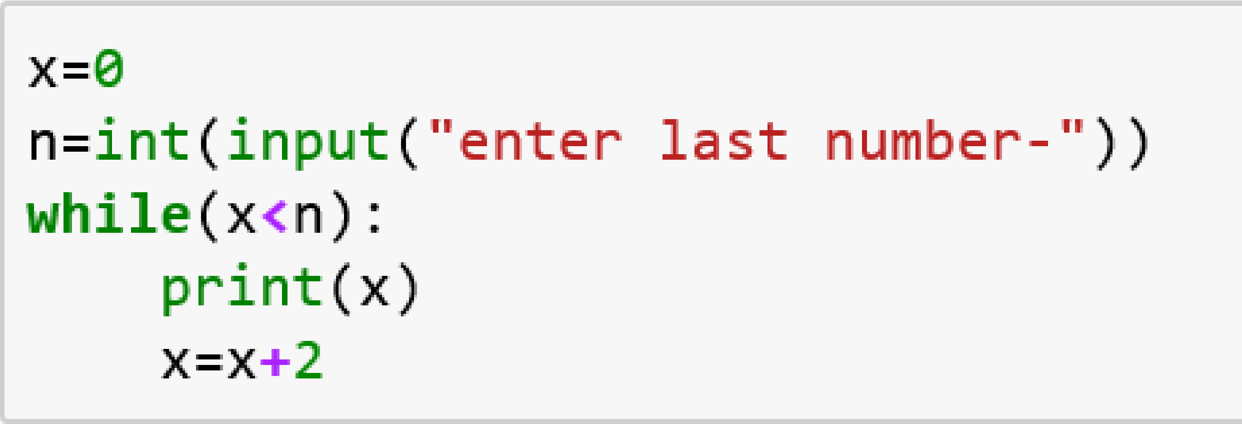


while statement	Example use

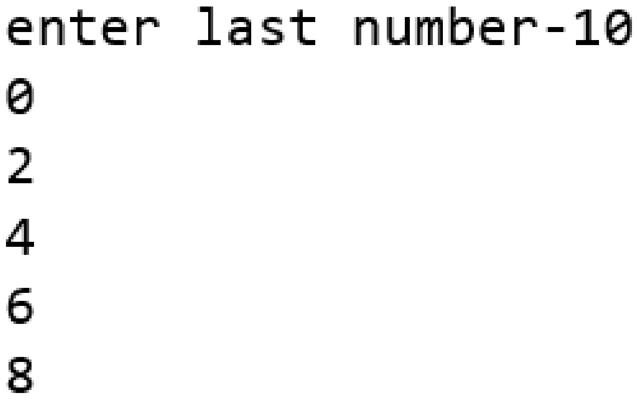
`while` *`condition:`*
    `suite`
	`sum = 0`
	`current = 1`
	
	`n = int(input('Enter value: '))`
	
	`while current <= n:`
	    `sum = sum + current`
	    `current = current + 1`

Example 1

- **Find all even numbers from 0 to n. where, n is given by user.**



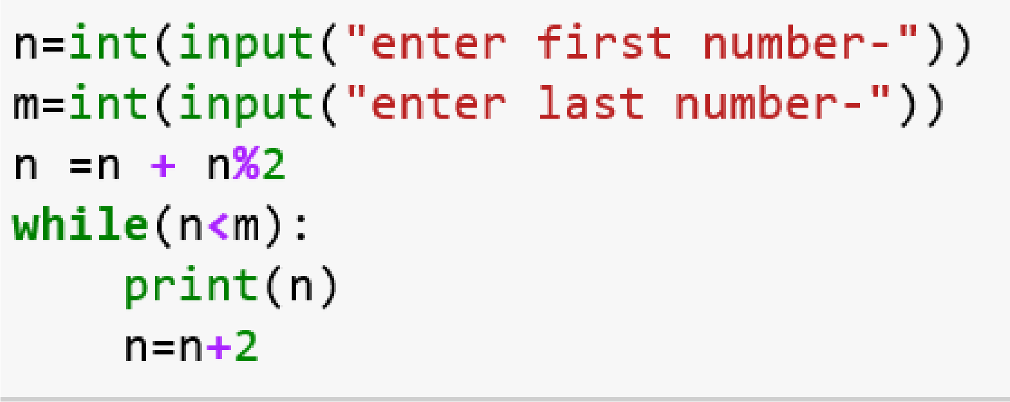
```
x=0
n=int(input("enter last number-"))
while(x<n):
    print(x)
    x=x+2
```



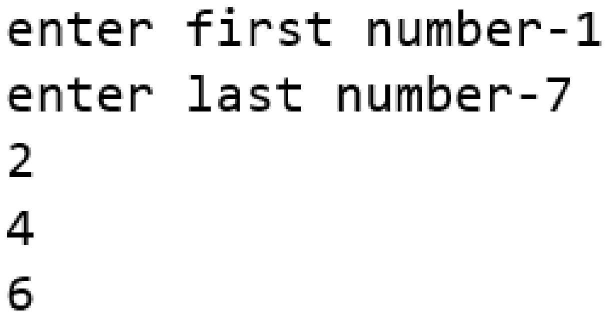
```
enter last number-10
0
2
4
6
8
```

Example 2

- **Print all even numbers between n to m. m should be greater than n.**



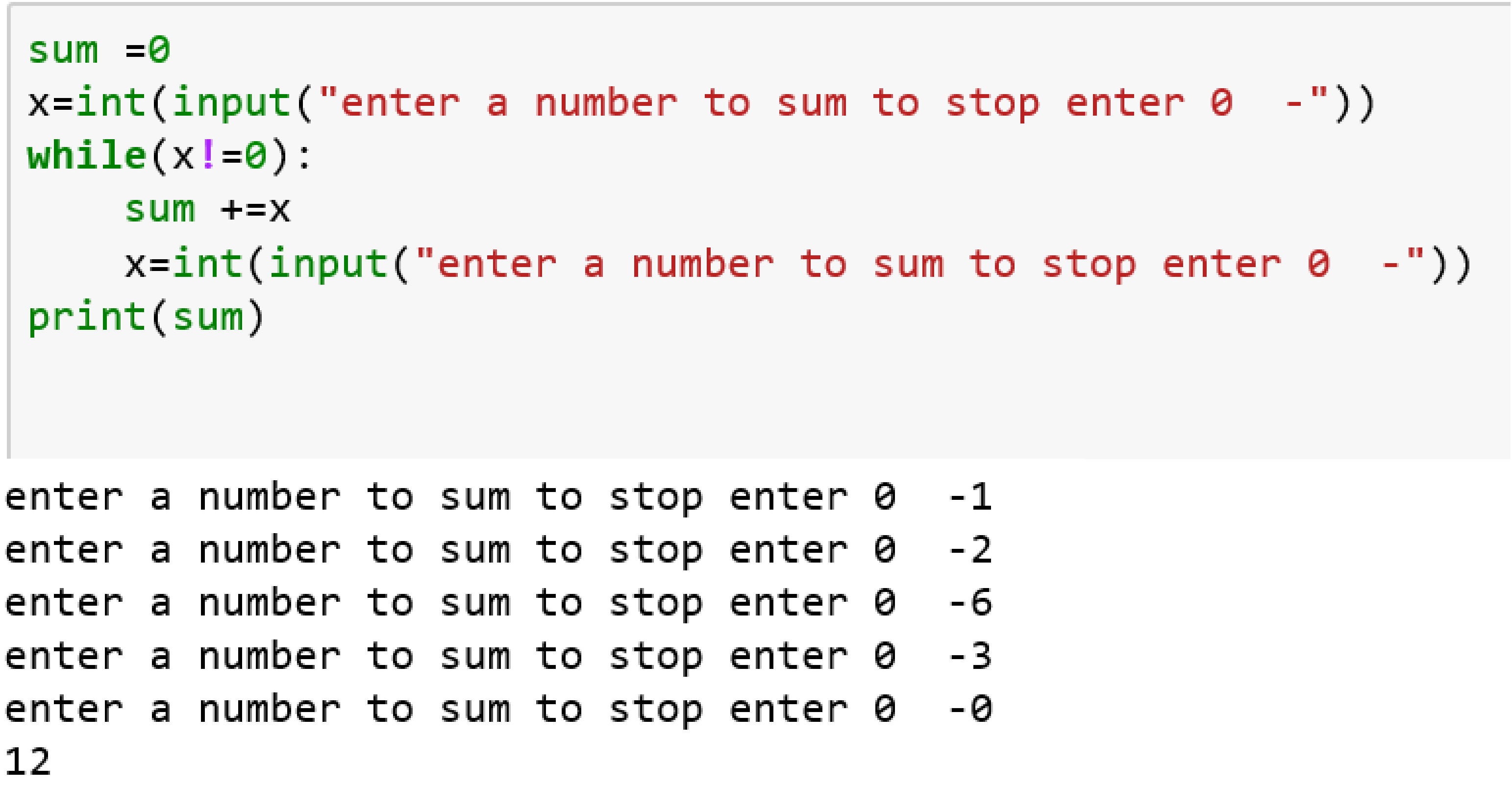
```python
n=int(input("enter first number-"))
m=int(input("enter last number-"))
n =n + n%2
while(n<m):
    print(n)
    n=n+2
```



```
enter first number-1
enter last number-7
2
4
6
```

Example 3

Write a program to take numbers from the user until he enter 0 as input. then print sum of all entered number.

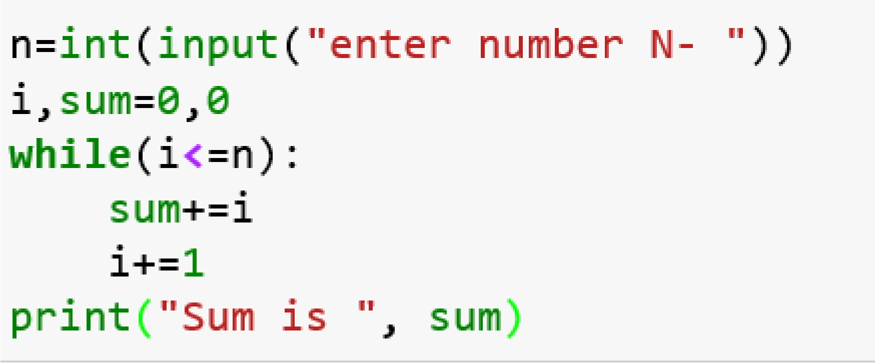


```
sum =0
x=int(input("enter a number to sum to stop enter 0  -"))
while(x!=0):
    sum +=x
    x=int(input("enter a number to sum to stop enter 0  -"))
print(sum)
```

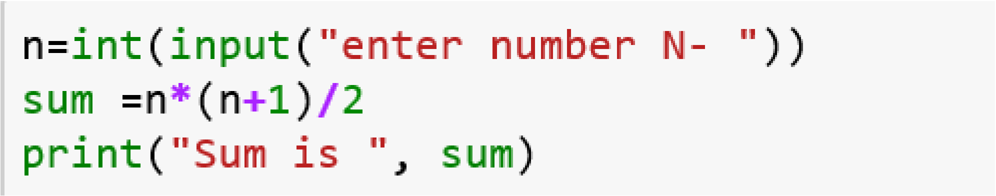
```
enter a number to sum to stop enter 0  -1
enter a number to sum to stop enter 0  -2
enter a number to sum to stop enter 0  -6
enter a number to sum to stop enter 0  -3
enter a number to sum to stop enter 0  -0
12
```

Example 4

- **Write an efficient program to determine sum of N natural numbers where N is given by user.**



```
n=int(input("enter number N- "))
i,sum=0,0
while(i<=n):
    sum+=i
    i+=1
print("Sum is ", sum)
```



```
n=int(input("enter number N- "))
sum =n*(n+1)/2
print("Sum is ", sum)
```

For Loop

(definite loop)

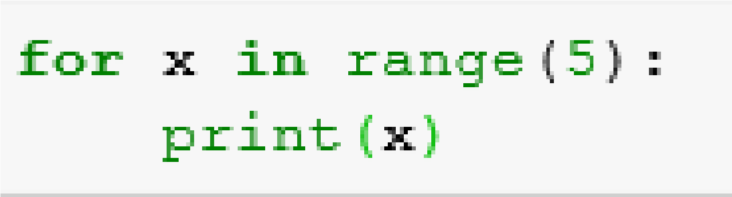
Loop to read sequence

- A **for** loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string). Can execute a set of statements, once for each item in a list, tuple, set etc.

Syntax:

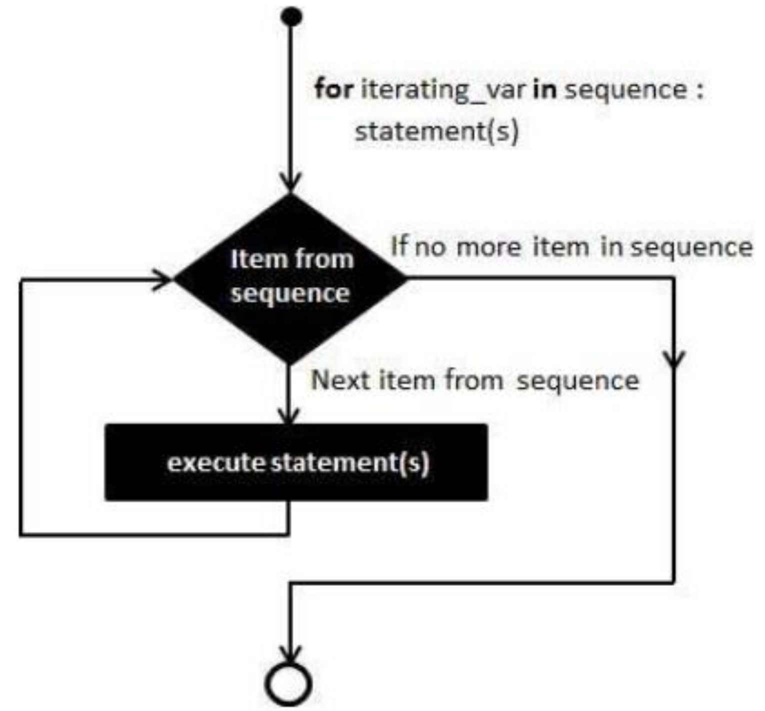
```
for iterating_var in sequence:
  statements(s)
```

- **Example:**



```
for x in range(5):
    print(x)
```

```
0
1
2
3
4
```



Sequences

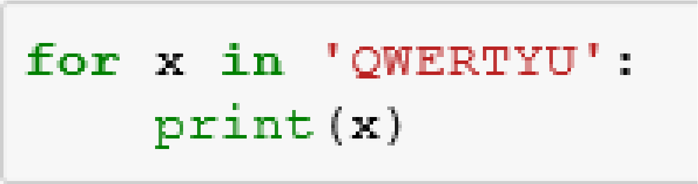
- Sequence of character - 'QWERTYUIOPASDFGHJKL’
- Sequence of words - ['abc','def','efg','ijk’]
- Sequence of numbers - [1,2,3,4,5,6,7,8,9]
- Sequence of mix data – [‘Suvi’, 4, “LKG”, “Bennett University”, 98.5

Sequence of numbers can also be generated as:

- range(start, end, difference)
- range(3)   = (0,1,2)
- range(1,5) = (1,2,3,4)
- range(3,9,2) = (3, 5, 7)
- range(9,2,-1) = (9,8,7,6,5,4,3)
- range(9,2,1) = []

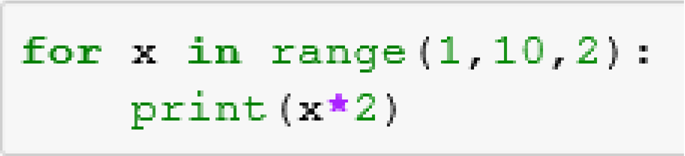
For Loop: What will be the output?

1.



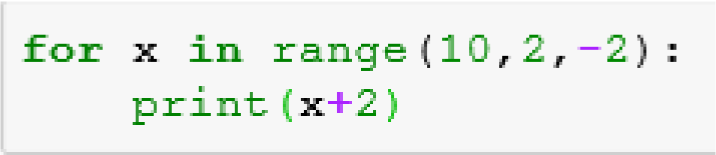
```
for x in 'QWERTYU':
    print(x)
```

2.



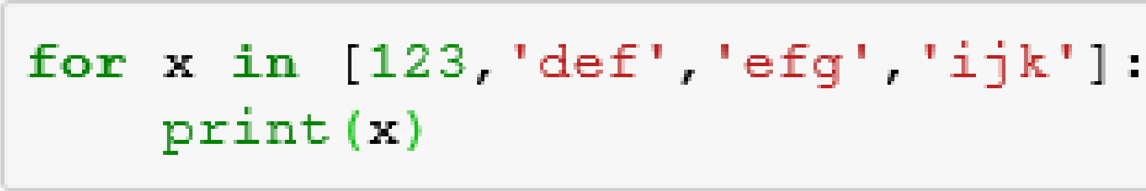
```
for x in range(1,10,2):
    print(x*2)
```

3.



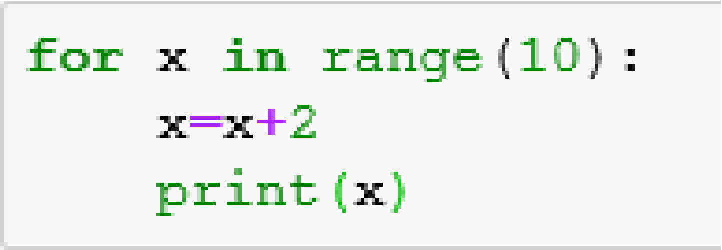
```
for x in range(10,2,-2):
    print(x+2)
```

4.



```
for x in [123,'def','efg','ijk']:
    print(x)
```

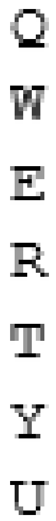
5.



```
for x in range(10):
    x=x+2
    print(x)
```

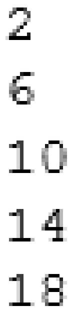
For Loop: Answers to Previous Questions

1



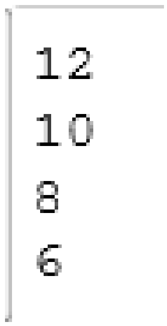
```
Q
W
E
R
T
Y
U
```

2



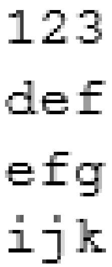
```
2
6
10
14
18
```

3



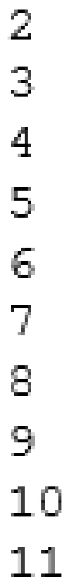
```
12
10
8
6
```

4



```
123
def
efg
ijk
```

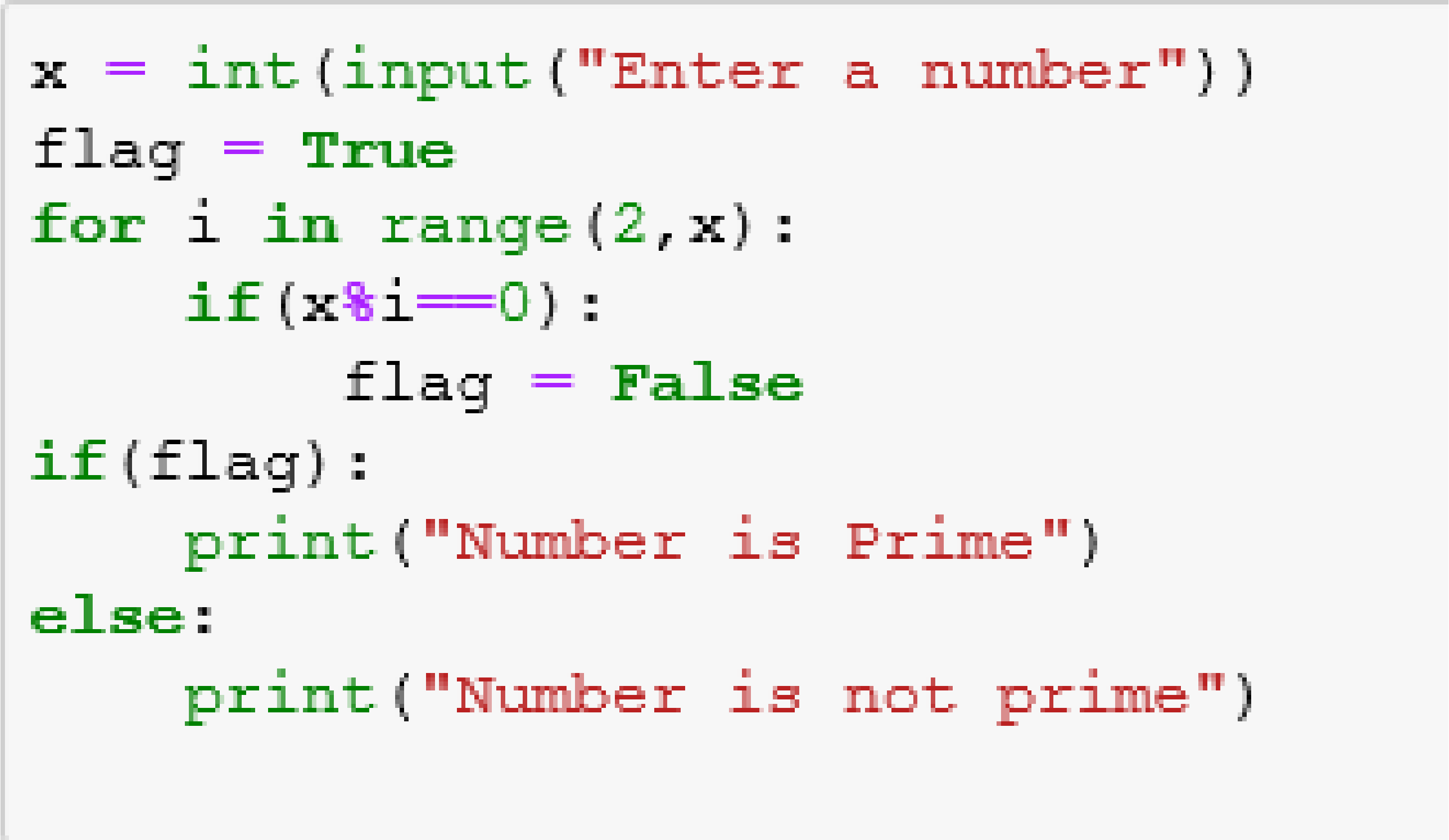
5



```
2
3
4
5
6
7
8
9
10
11
```

Exercise: Write a program to find whether a given number is prime or not

- **Solution:**



```python
x = int(input("Enter a number"))
flag = True
for i in range(2,x):
    if(x%i==0):
        flag = False
if(flag):
    print("Number is Prime")
else:
    print("Number is not prime")
```

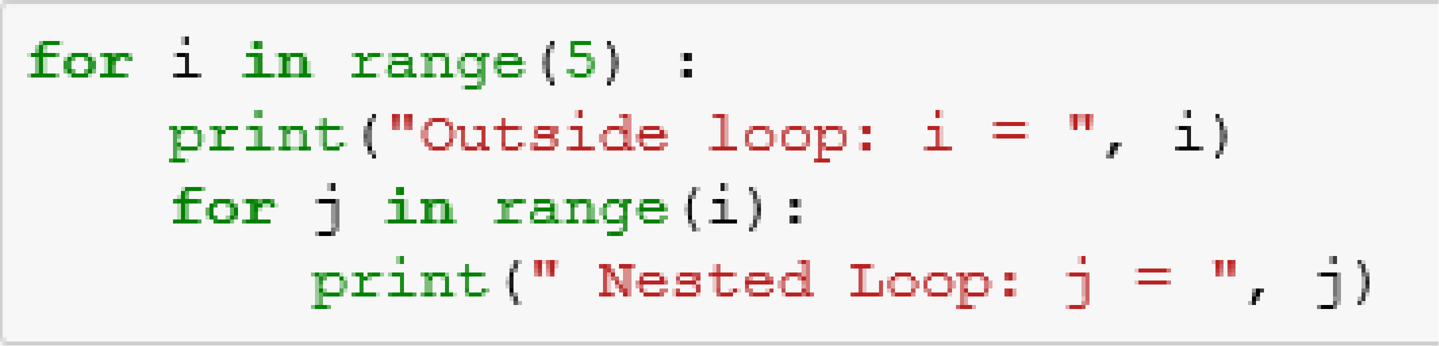
Problem Exercise 2

- Let one grain of wheat be placed on the first square of a chessboard, two grains on the second square, four grains on the third square, eight grains on the fourth square, and so on until all square are filled in chessboard. what will be the total weight in ton of grains on whole 8×8 chessboard? If 15432 grains in one kg and 907.18 kg in one ton then.

Nested Loop

- Loop inside a loop a is called nested loop.

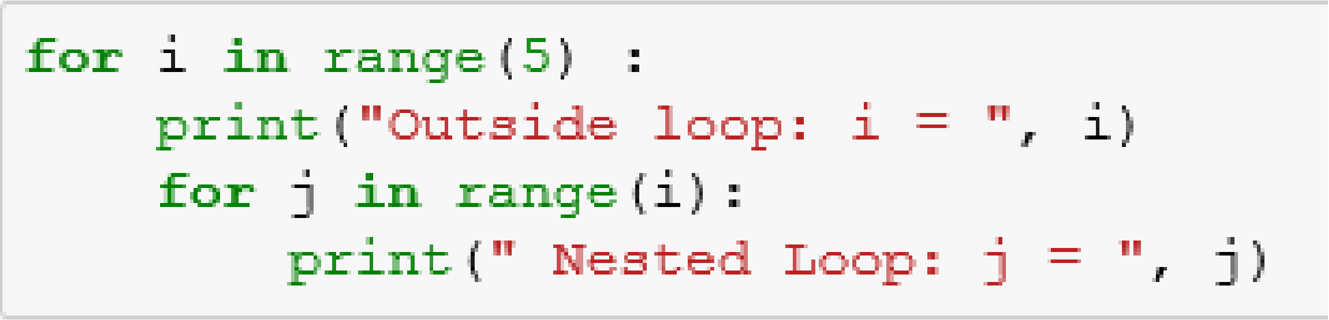
- **Example:**



```
for i in range(5) :
    print("Outside loop: i = ", i)
    for j in range(i):
        print(" Nested Loop: j = ", j)
```

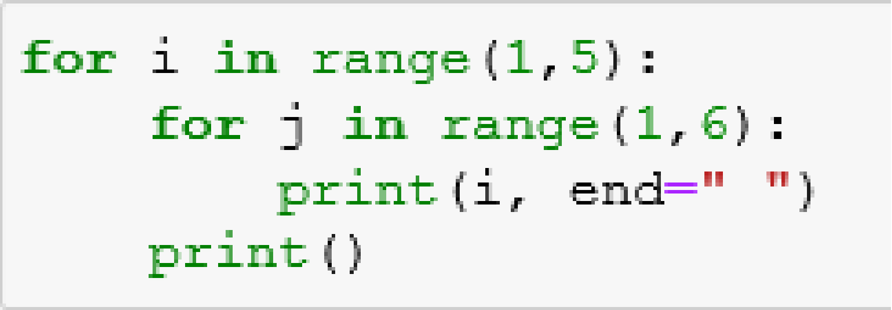
Nested Loop: What will be the output?

1.



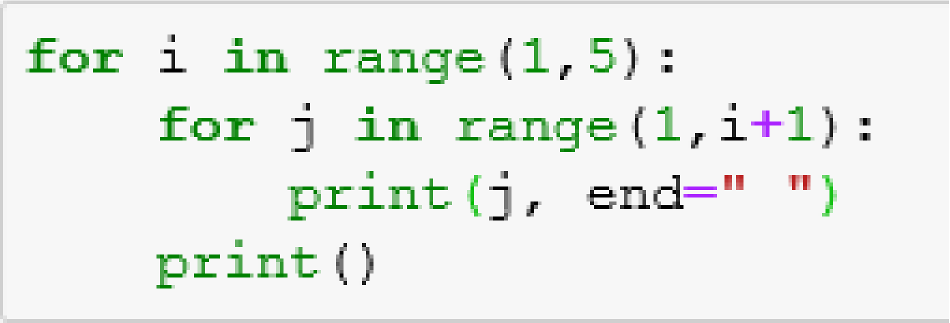
```
for i in range(5) :
    print("Outside loop: i = ", i)
    for j in range(i):
        print(" Nested Loop: j = ", j)
```

2.



```
for i in range(1,5):
    for j in range(1,6):
        print(i, end=" ")
    print()
```

3.

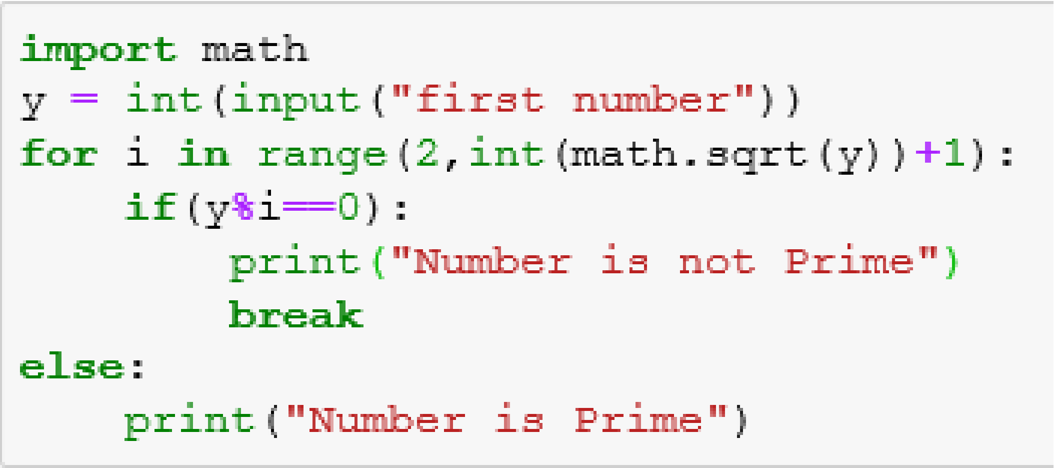


```
for i in range(1,5):
    for j in range(1,i+1):
        print(j, end=" ")
    print()
```

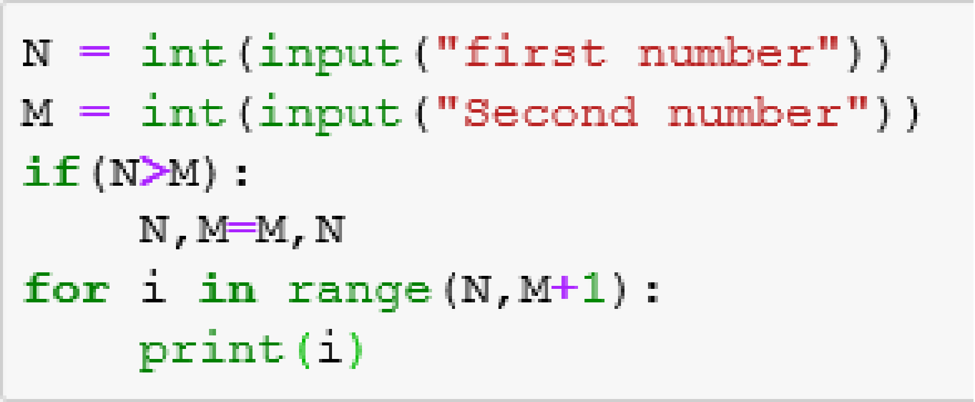
4.

5.

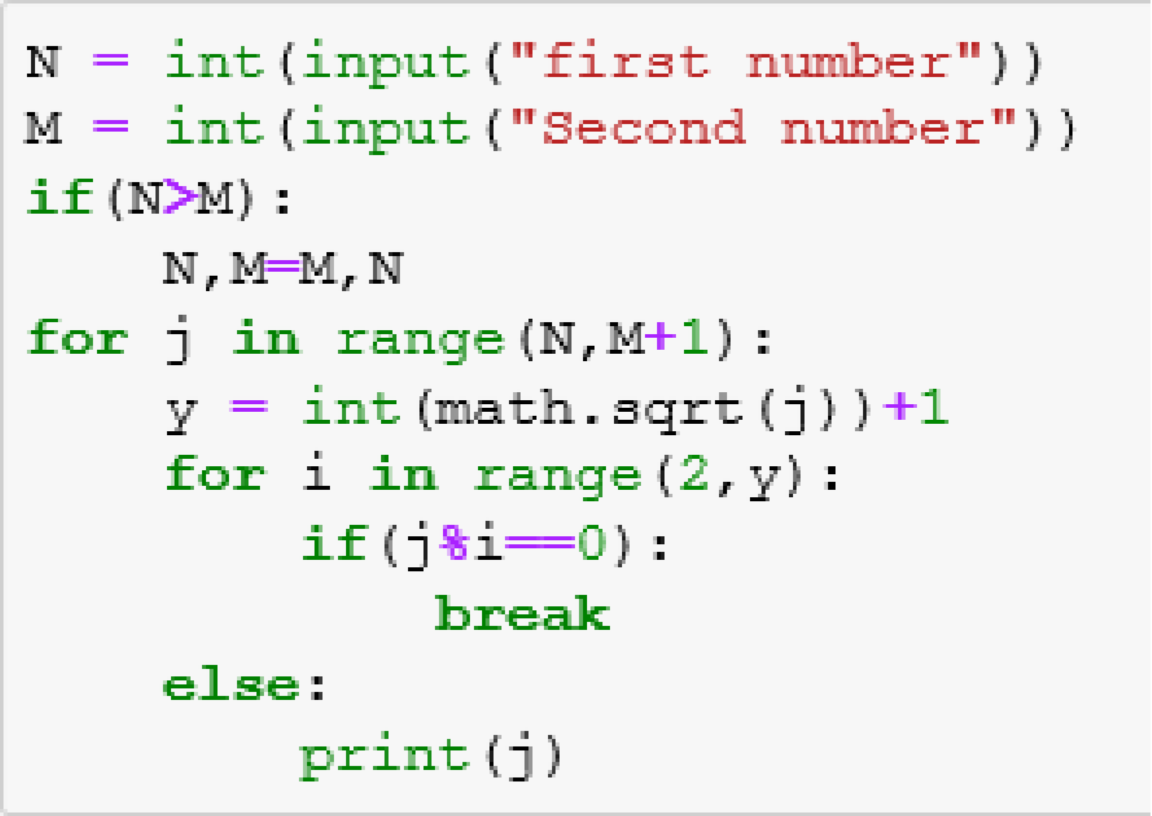
Find all prime numbers between given two numbers



```
import math
y = int(input("first number"))
for i in range(2,int(math.sqrt(y))+1):
    if(y%i==0):
        print("Number is not Prime")
        break
else:
    print("Number is Prime")
```

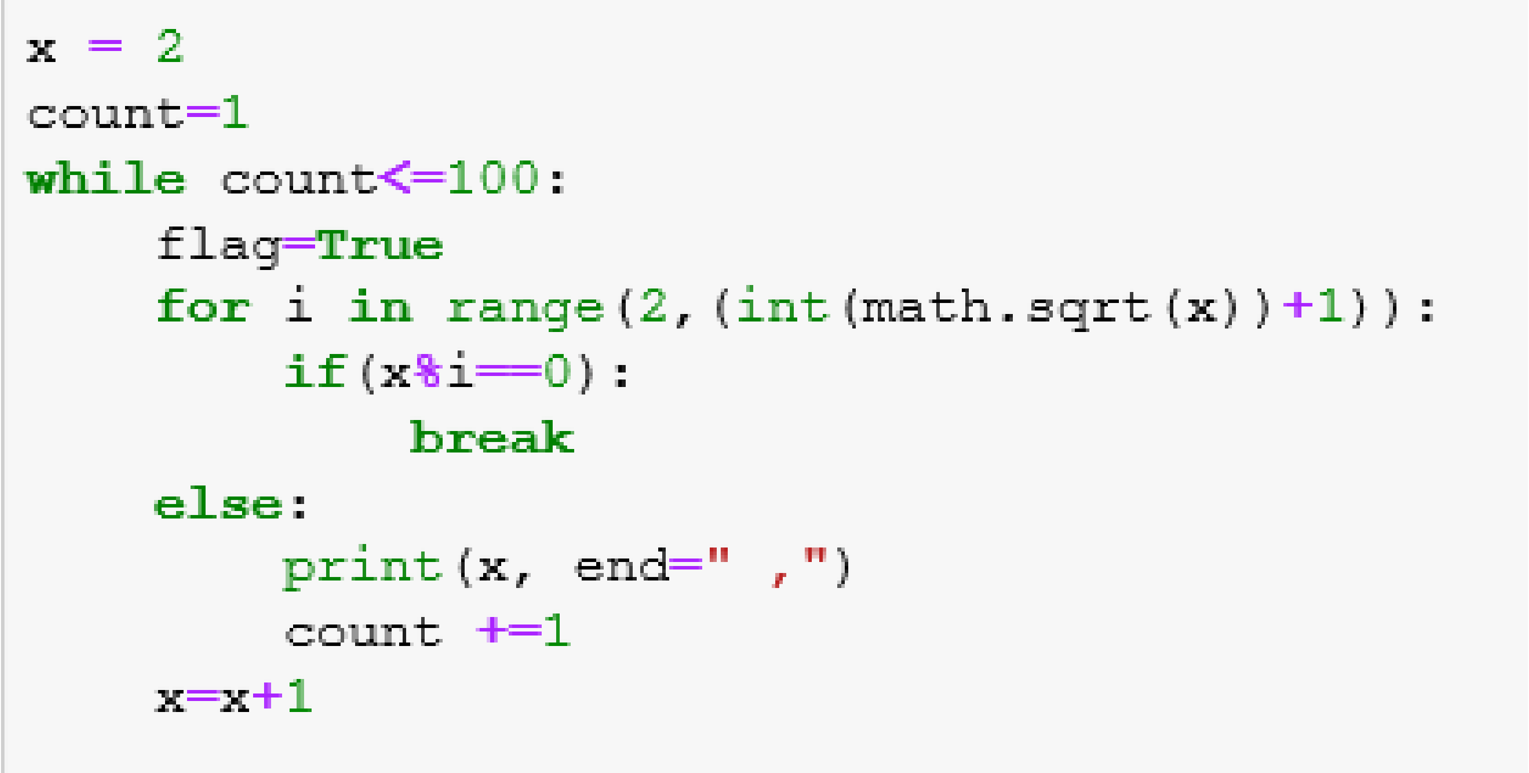


```
N = int(input("first number"))
M = int(input("Second number"))
if(N>M):
    N,M=M,N
for i in range(N,M+1):
    print(i)
```



```
N = int(input("first number"))
M = int(input("Second number"))
if(N>M):
    N,M=M,N
for j in range(N,M+1):
    y = int(math.sqrt(j))+1
    for i in range(2,y):
        if(j%i==0):
            break
    else:
        print(j)
```

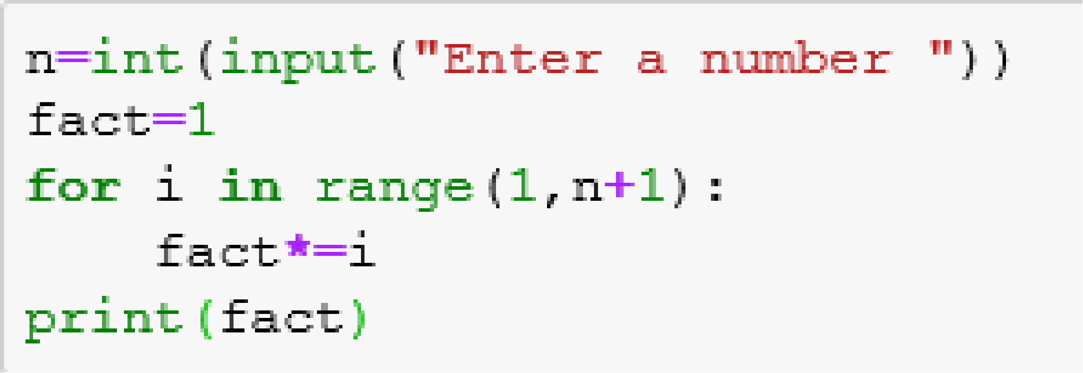
Find first 100 prime numbers start from 2.



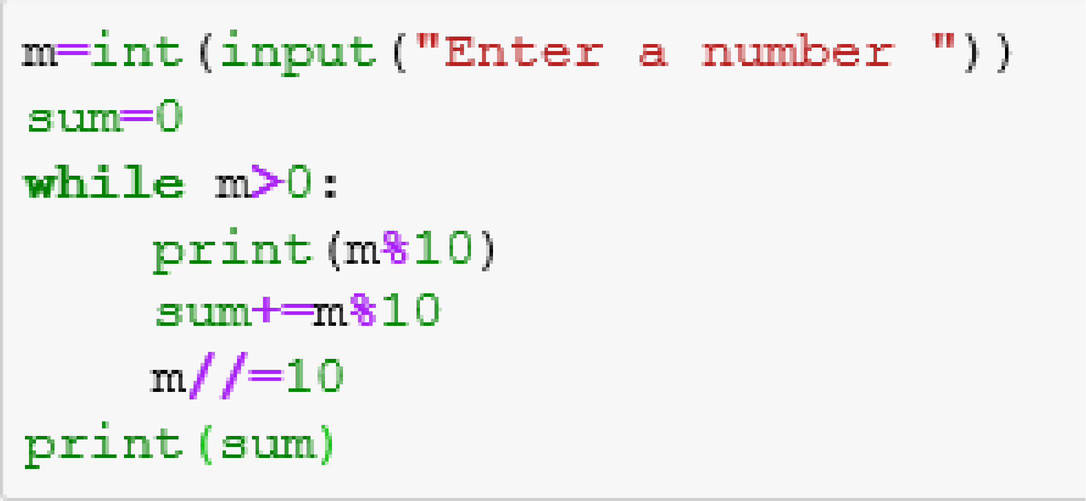
```
x = 2
count=1
while count<=100:
    flag=True
    for i in range(2,(int(math.sqrt(x))+1)):
        if(x%i==0):
            break
    else:
        print(x, end=" ,")
        count +=1
    x=x+1
```

Find number is Strong or not

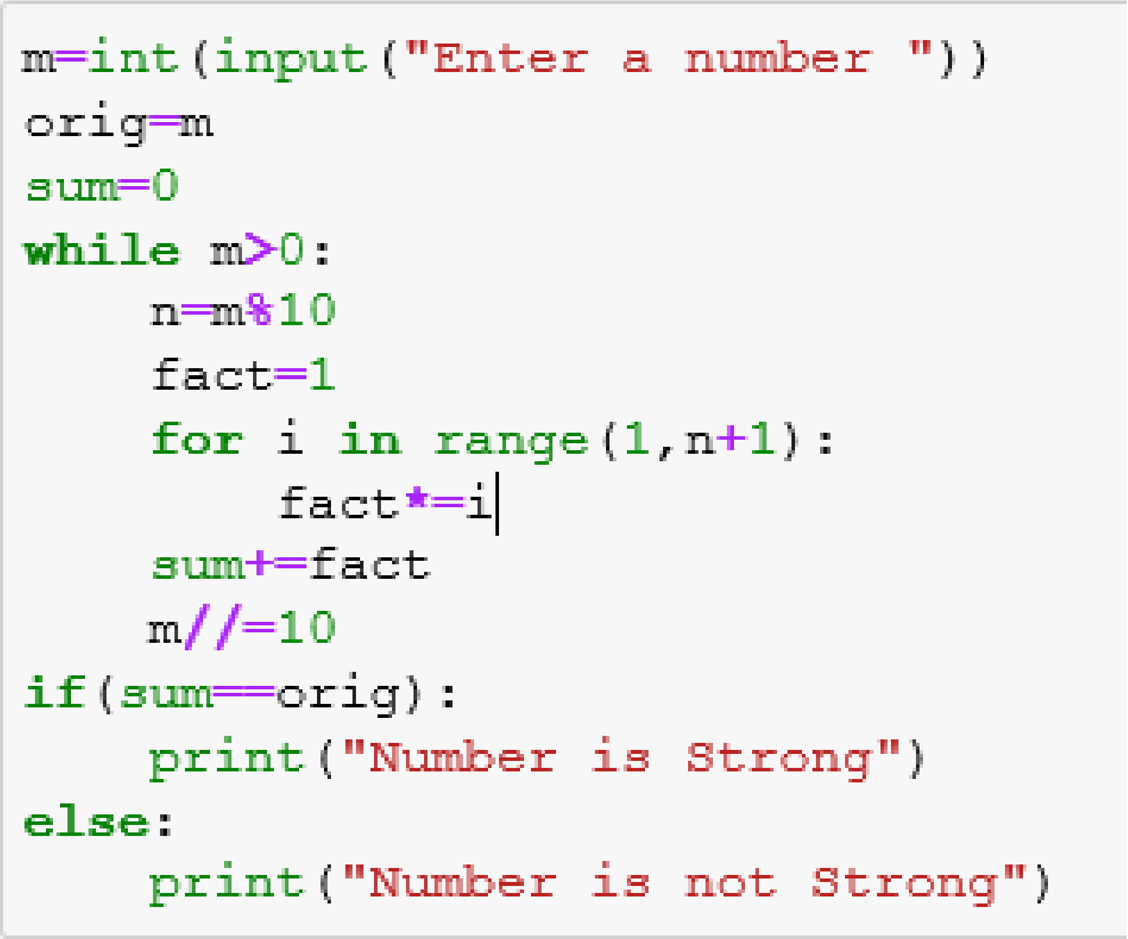
If the sum of the factorial of the digits in a number is equal to the original number, the number is a strong number.



```
n=int(input("Enter a number "))
fact=1
for i in range(1,n+1):
    fact*=i
print(fact)
```

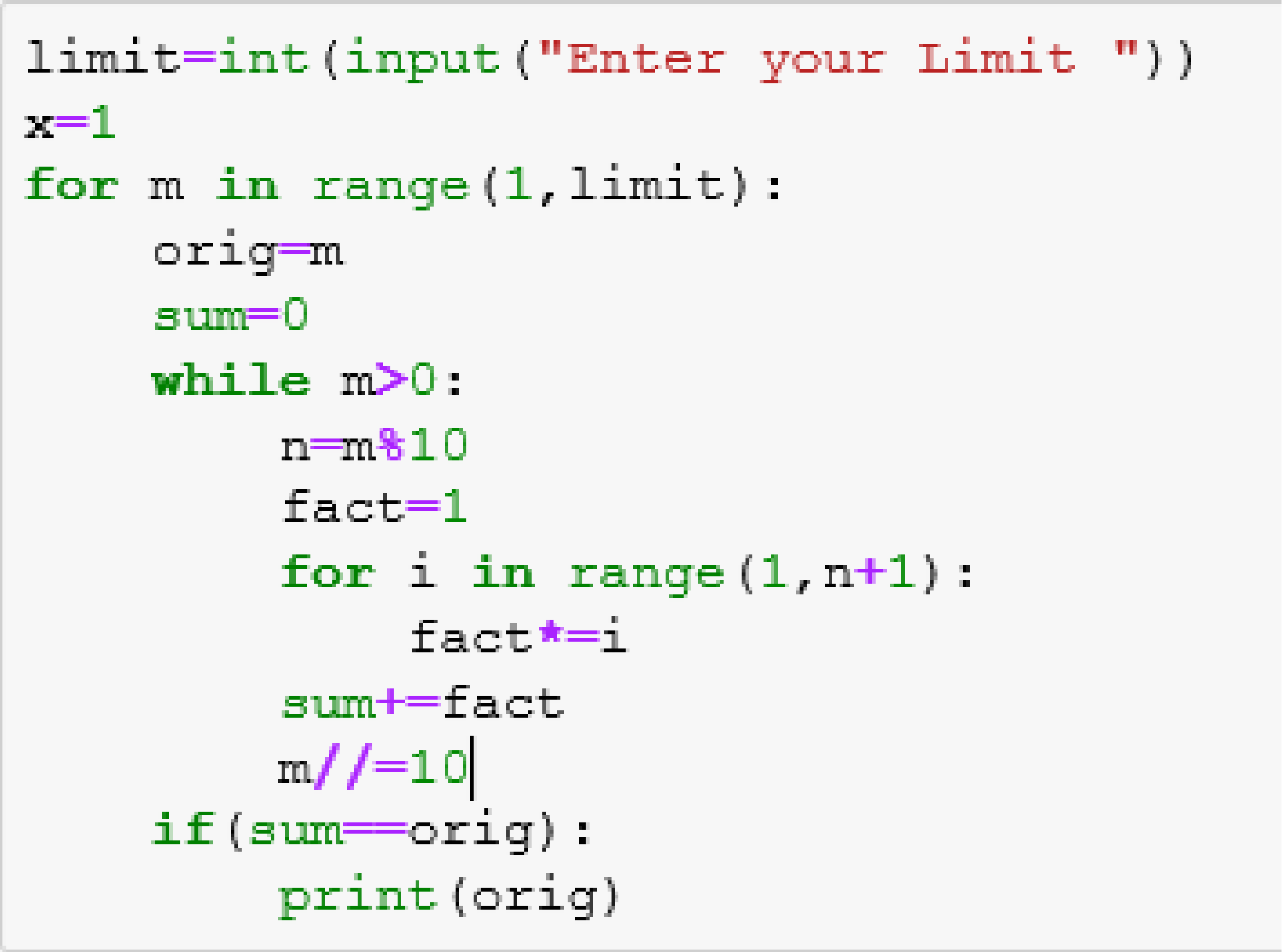


```
m=int(input("Enter a number "))
sum=0
while m>0:
    print(m%10)
    sum+=m%10
    m//=10
print(sum)
```



```
m=int(input("Enter a number "))
orig=m
sum=0
while m>0:
    n=m%10
    fact=1
    for i in range(1,n+1):
        fact*=i
    sum+=fact
    m//=10
if(sum==orig):
    print("Number is Strong")
else:
    print("Number is not Strong")
```

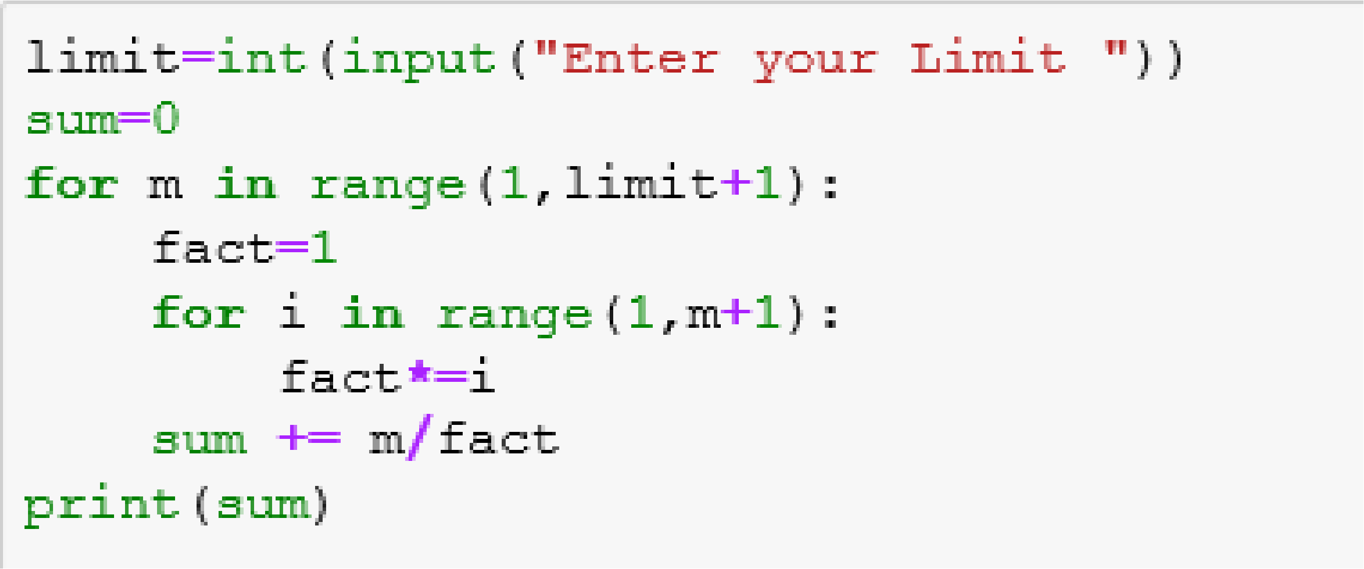
Accept the limit and print the strong numbers from 1 to the given limit.



```
limit=int(input("Enter your Limit "))
x=1
for m in range(1,limit):
    orig=m
    sum=0
    while m>0:
        n=m%10
        fact=1
        for i in range(1,n+1):
            fact*=i
        sum+=fact
        m//=10
    if(sum==orig):
        print(orig)
```

Calculate and print the sum of following series

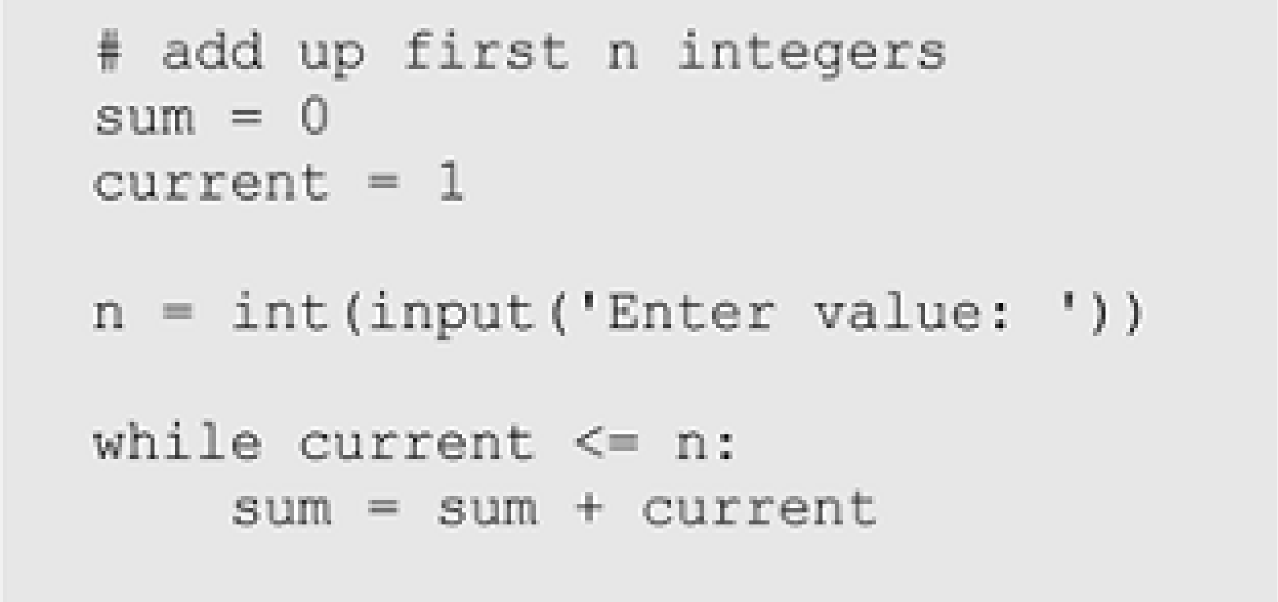
$$\frac{1}{1!} + \frac{2}{2!} + \frac{3}{3!} + \frac{4}{4!} + \frac{5}{5!} + \frac{6}{6!} \,.......$$



```
limit=int(input("Enter your Limit "))
sum=0
for m in range(1,limit+1):
    fact=1
    for i in range(1,m+1):
        fact*=i
    sum += m/fact
print(sum)
```

Infinite loop

- An **infinite loop** is an iterative control structure that never terminates (or eventually terminates with a system error).
- Infinite loops are generally the result of programming errors.
- **For example:** if the condition of a while loop can never be false, an infinite loop will result when executed.



```
# add up first n integers
sum = 0
current = 1

n = int(input('Enter value: '))

while current <= n:
    sum = sum + current
```

Loop Control Statements

- **Break Statement:** Terminates the loop statement and transfers execution to the statement immediately following the loop.
- **Continue Statement:** Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

If x=5:
pass

- **Pass Statement:**The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

```
Use of pass in if:
a = 33
b = 200
if b > a:
  pass
```

```
Example:
for letter in 'Python’:
        if letter == 'h’:
             pass
             print('This is pass block’)
        print('Current Letter :', letter)
print(“Loop Ended!“)
```

```
Output:
Current Letter : P
Current Letter : y
Current Letter : t
This is pass block
Current Letter : h
Current Letter : o
Current Letter : n
Loop Ended
```

Break and Continue: Examples

- **Break Statement:**

```
Ex.1: fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)                        Output:
  if x == "banana":               apple
    break                         banana
```

```
Ex.2: fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

- **Continue Statement:**

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

MCQs

1. A while loop continues to iterate until its condition becomes false.
   - [✓ a) TRUE]
   - b) FALSE
2. A while loop executes zero or more times.
   - [✓ a) TRUE]
   - b) FALSE
3. All iteration can be achieved by a while loop.
   - [✓ a) TRUE]
   - b) FALSE
4. An infinite loop is an iterative control structures that,
   - a) Loops forever and must be forced to terminate
   - b) Loops until the program terminates with a system error
   - [✓ c) Both of the above]
5. The terms definite loop and indefinite loop are used to indicate whether,
   - a) A given loop executes at least once
   - [✓ b) The number of times that a loop is executed can be determined before the loop is executed.]
   - c) Both of the above
6. A Boolean flag is,
   - [✓ a) A variable]
   - [✓ b) Has the value True or False]
   - [✓ c) Is used as a condition for control statements]
   - [✓ d) All of the above]

MCQs: Answers

1. A while loop continues to iterate until its condition becomes false.
   - **a) TRUE**
   - b) FALSE
2. A while loop executes zero or more times.
   - **a) TRUE**
   - b) FALSE
3. All iteration can be achieved by a while loop.
   - **a) TRUE**
   - b) FALSE
4. An infinite loop is an iterative control structures that,
   - a) Loops forever and must be forced to terminate
   - b) Loops until the program terminates with a system error
   - **c) Both of the above**
5. The terms definite loop and indefinite loop are used to indicate whether,
   - a) A given loop executes at least once
   - **b) The number of times that a loop is executed can be determined before the loop is executed.**
   - c) Both of the above
6. A Boolean flag is,
   - a) A variable
   - b) Has the value True or False
   - c) Is used as a condition for control statements
   - **d) All of the above**