**Flow Control**

**if...else statement in Python**

Decision making is required when we want to execute a code only if a certain condition is satisfied. The *if…elif…else* statement is used in Python for decision making.

**Python if Statement Syntax**

if test expression:

statement(s)

Here, the program evaluates the test expression and will execute statement(s) only if the text expression is True. If the text expression is False, the statement(s) is not executed.

In Python, the body of the if statement is indicated by the indentation. Body starts with an indentation and the first unindented line marks the end. Python interprets non-zero values as True. None and 0 are interpreted as False.

Example:

num = 3

if num > 0:

print(num, "is a positive number.")

print("This is always printed.")

num = -1

if num > 0:

print(num, "is a positive number.")

print("This is also always printed.")

## Python if...else Statement

### Syntax of if...else

if test expression:

Body of if

else:

Body of else

The if..else statement evaluates test expression and will execute body of if only when test condition is True.

If the condition is False, body of else is executed. Indentation is used to separate the blocks.

Example:

if num >= 0:

print("Positive or Zero")

else:

print("Negative number")

## Python if...elif...else Statement

### Syntax of if...elif...else

if test expression:

Body of if

elif test expression:

Body of elif

else:

Body of else

The elif is short for else if. It allows us to check for multiple expressions. If the condition for if is False, it checks the condition of the next elif block and so on.

If all the conditions are False, body of else is executed. Only one block among the several if...elif...else blocks is executed according to the condition. The if block can have only one else block. But it can have multiple elif blocks.

Example :

num = 3.4

if num > 0:

print("Positive number")

elif num == 0:

print("Zero")

else:

print("Negative number")

**Loops**

**for loop in Python**

The for loop in Python is used to iterate over a sequence ([list](https://www.programiz.com/python-programming/list), [tuple](https://www.programiz.com/python-programming/tuple), [string](https://www.programiz.com/python-programming/string)) or other iterable objects. Iterating over a sequence is called traversal.

**Syntax of for Loop**

for *val* in *sequence*:

Body of for

Here, val is the variable that takes the value of the item inside the *sequence* on each iteration.

Loop continues until we reach the last item in the sequence. The body of *for* loop is separated from the rest of the code using indentation.

Example:

numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]

# variable to store the sum

sum = 0

# iterate over the list

for val in numbers:

sum = sum+val

# Output: The sum is 48

print("The sum is", sum)

## The range() function

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

We can also define the start, stop and step size as range(start,stop,step size). *step size* defaults to 1 if not provided.

This function does not store all the values in memory, it would be inefficient. So it remembers the start, stop, step size and generates the next number on the go.

To force this function to output all the items, we can use the function list().

print(range(10))

# Output: range(0, 10)

print(list(range(10)))

# Output: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

print(list(range(2, 8)))

# Output: [2, 3, 4, 5, 6, 7]

print(list(range(2, 20, 3)))

# Output: [2, 5, 8, 11, 14, 17]

Example: digits = [0, 1, 5]

for i in digits:

print(i)

## while loop in Python

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 beforehand, the number of times to iterate.

### Syntax of while Loop in Python

while test\_expression:

Body of while

In *while* loop, test expression is checked first. The body of the loop is entered only if the test\_expression evaluates to True. After one iteration, the test expression is checked again. This process continues until the test\_expression evaluates to False.

In Python, the body of the while loop is determined through indentation.

Body starts with indentation and the first unindented line marks the end.

Python interprets any non-zero value as True. None and 0 are interpreted as False.

Example:

n = 10

# initialize sum and counter

sum = 0

i = 1

while i <= n:

sum = sum + i

i = i+1 # update counter

# print the sum

print("The sum is", sum)

list1=[1,5,6,8,6,11,23,5]

name='balu'

for val in range(0,11):

print(val)

n=11

i=1

while i<n:

print(i)

i=i+1

///////////////////////////////////

*Infinite loop* – If loop continues to execute infinitely.

Example:

while True:

print(‘hello’)

//////////////////////////////////////////

## Break and Continue in Python

In Python, *break* and *continue* statements can alter the flow of a normal loop.

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

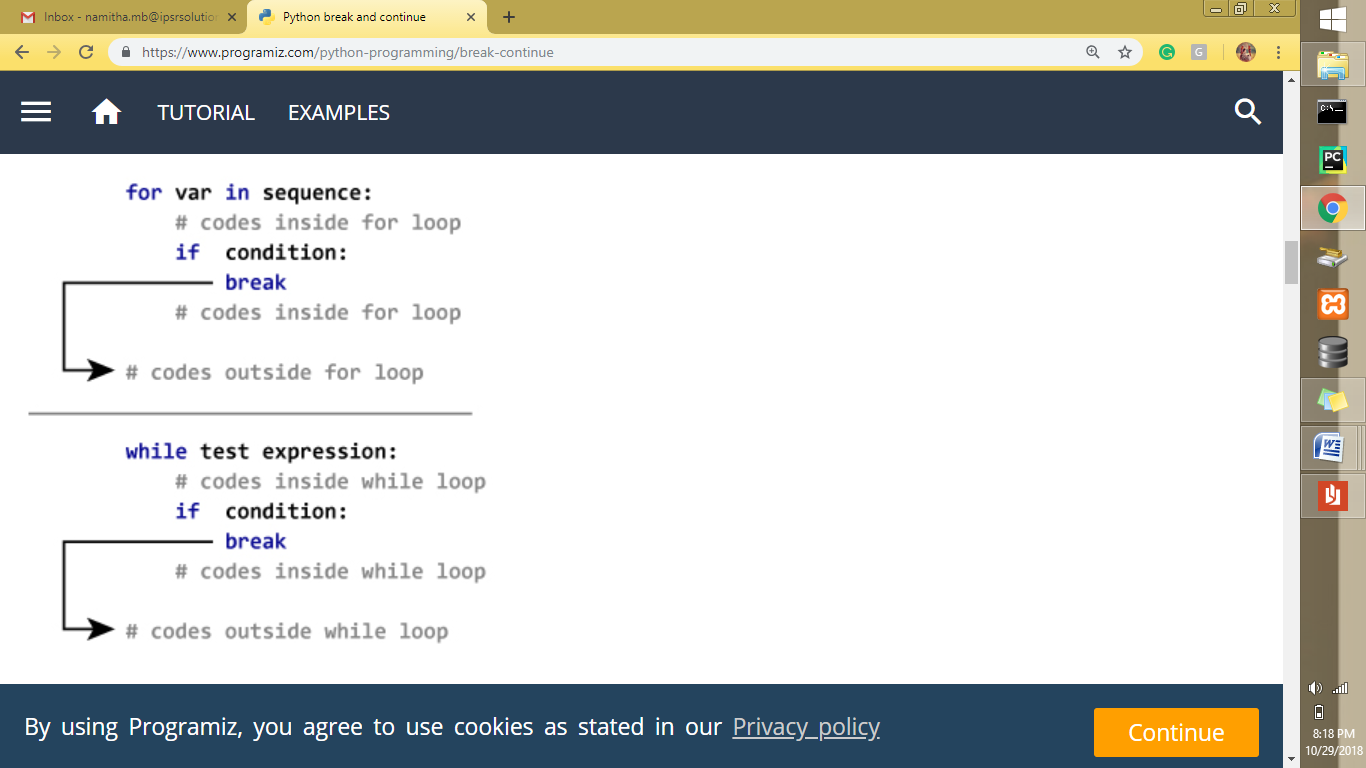
The break and continue statements are used in these cases.

## Python 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.

### Syntax of break

break



Example:

for val in "string":

if val == "i":

break

print(val)

print("The end")

## Python 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.

### Syntax of Continue

Continue



Example:

for val in "string":

if val == "i":

continue

print(val)

print("The end")

**Example:**

for i in range(10):

print(i, end=" ")

else: # This ‘else’ will be executed whenever there is no ‘break’ in ‘for’ loop

print()

print("odd")

**Result:**

0 1 2 3 4 5 6 7 8 9

odd

**Example:**

for i in range(1, 4):

print(i)

break

else: # Not executed as there is a 'break' in 'for'

print("No Break")

**Result:**

1

**Example:**

for i in range(1, 4):

print(i)

continue

print(i)

else:

print("No Break")

**Result:**

1

2

3

No Break

**Example:**

c = 0

while (c < 1):

    c = c+1

    print(c)

    break

else:

    print("No Break")

**Result:**

1

**Example:**

words = ['cat', 'window', 'defenestrate']

for w in words:

print(w, len(w))

**Result:**

cat 3

window 6

defenestrate 12

**Example:**

import math

a = [56.2, float('NaN'), 51.7, 55.3, 52.5, float('NaN'), 47.8]

b = []

for v in a:

if not math.isnan(v):

b.append(v)

print(b)

**Result:**

[56.2, 51.7, 55.3, 52.5, 47.8]

**Example:**

import math

a = [56.2, float('NaN'), 51.7, 55.3, 52.5, float('NaN'), 47.8]

b = []

for v in a:

if math.isnan(v):

b.append(v)

print(b)

**Result:**

[nan, nan]

**Example:**

print(sum(range(4)))

**Result:**

6

//////////////////////////////////////////

The *pass* statement will do nothing. It can be used when a statement is required syntactically, but the program requires no action.

/////////////////////////////////////////

**Example:**

while True: # It will do nothing. Keyboard interrupt (cntrl+c) will exit the program

pass

**Example:**

sequence = {'p', 'a', 's', 's'}

for val in sequence:

pass

**Example:**

def function(args):

pass

**Example:**

class hello:

pass

////////////////////////////////////////

**Example:**

#Python program to check if a number is positive, negative or 0. Using ‘if…elif…else’.

num = float(input("Enter a number: "))

if num > 0:

print("Positive number")

elif num == 0:

print("Zero")

else:

print("Negative number")

**Result:**

Enter a number: 12

Positive number

**Example:**

## #Python program to check if a number is positive, negative or 0. Using nested ‘if’.

num = float(input("Enter a number: "))

if num >= 0:

if num == 0:

print("Zero")

else:

print("Positive number")

else:

print("Negative number")

**Result:**

Enter a number: 0

Zero

**Example:**

# python program to check if a number is odd or even.

num = int(input("Enter a number: "))

if (num % 2) == 0:

print(num,"is Even")

else:

print(num,"is Odd")

**Result:**

Enter a number: 11

11 is Odd

**Example:**

#Python program to find H.C.F of two numbers. Using loop.

x = 18

y = 12

if x > y:

smaller = y

else:

smaller = x

for i in range(1, smaller+1):

if((x % i == 0) and (y % i == 0)):

hcf = i

print(hcf)

**Result:**

6

/////////////////////////////

**Euclidean algorithm**

This algorithm is based on the fact that H.C.F. of two numbers divides their difference as well.

In this algorithm, we divide the greater by smaller and take the remainder. Now, divide the smaller by this remainder. Repeat until the remainder is 0.

For example, if we want to find the H.C.F. of 54 and 24, we divide 54 by 24. The remainder is 6. Now, we divide 24 by 6 and the remainder is 0. Hence, 6 is the required H.C.F.

///////////////////////////

**Example:**

## #Python program to find H.C.F of two numbers. Using Euclidean Algorithm.

## x = 300

## y = 400

## while(y):

## x, y = y, x%y

## print(x)

**Result:**

100

**Example of swapping variable:**

x = 5

y = 10

temp = x

x = y

y = temp

print('The value of x after swapping:',x)

print('The value of y after swapping:',y)

**Result:**

The value of x after swapping: 10

The value of y after swapping: 5

**Example of swapping variable without using temporary variable:**

x = 5

y = 10

x, y = y, x

print("x =", x)

print("y =", y)

**Result:**

x = 10

y = 5

**Example of swapping variable by using arithmetic operators:**

x = 5

y = 10

x = x + y

y = x - y

x = x - y

print("x =", x)

print("y =", y)

**Result:**

x = 10

y = 5

**Example of swapping variable by using arithmetic operators:**

x = 24.54

y = 31.22

x = x \* y

y = x / y

x = x / y

print("x =", x)

print("y =", y)

print("x =", int(x))

print("y =", int(y))

**Result:**

x = 31.22

y = 24.54

x = 31

y = 24

**Example of swapping variable by using XOR (This will work for integers only):**

x = 24

print(bin(x))

y = 31

print(bin(y))

x = x ^ y

print(bin(x))

y = x ^ y

print(bin(y))

x = x ^ y

print(bin(x))

print("x =", x)

print("y =", y)

**Result:**

0b11000

0b11111

0b111

0b11000

0b11111

x = 31

y = 24

//////////////////////////////////////

A leap year is exactly divisible by 4. A leap year like ‘2004’ is divisible by 4, but can’t be divided exactly by 100. The century year is a leap year only if it is perfectly divisible by 400.

////////////////////////////////////

**Example:**

#Python program to check if year is a leap year or not

y = 2000

if (y % 4) == 0:

if (y % 100) == 0:

if (y % 400) == 0:

print(y,"is a leap year")

else:

print(y,"is not a leap year")

else:

print(y,"is a leap year")

else:

print(y,"is not a leap year")

**Result:**

2000 is a leap year

**Example:**

# Python program to find the largest number among the three numbers.

num1 = float(input("Enter first number: "))

num2 = float(input("Enter second number: "))

num3 = float(input("Enter third number: "))

if (num1 >= num2) and (num1 >= num3):

largest = num1

elif (num2 >= num1) and (num2 >= num3):

largest = num2

else:

largest = num3

print("The largest number is", largest)

**Result:**

Enter first number: 22

Enter second number: 45

Enter third number: 91

The largest number is 91

**Example:**

# Program to check if a number is prime or not. Numbers less than or equal to 1 are not prime numbers.

num = int(input("Enter a number: "))

# prime numbers are greater than 1

if num > 1:

# check for factors

for i in range(2,num):

if (num % i) == 0:

print(num,"is not a prime number")

print(i,"times",num//i,"is",num)

break

else:

print(num,"is a prime number")

# if input number is less than or equal to 1, it is not prime

else:

print(num,"is not a prime number")

**Result:**

Enter a number: 79

79 is a prime number

**Example:**

# Program to find square root.

d = 81 \*\* 0.5

print(int(d))

**Result:**

9

**Example:**

d = 1728 \*\* (1/3)

print(round(d))

**Result:**

12

///////////////////////////

A composite number must have a factor less than the square root of that number. Otherwise, the number is prime.

//////////////////////////

**Example:**

# Program to check if a number is prime or not.

num = int(input("Enter a number: "))

# prime numbers are greater than 1

if num > 1:

# check for factors

for i in range(2,int(num \*\* 0.5)):

if (num % i) == 0:

print(num,"is not a prime number")

print(i,"times",num//i,"is",num)

break

else:

print(num,"is a prime number")

# if input number is less than or equal to 1, it is not prime

else:

print(num,"is not a prime number")

**Result:**

Enter a number: 13

13 is a prime number

**Example:**

# Python program to display all the prime numbers within an interval

lower = 900

upper = 1000

print("Prime numbers between", lower, "and", upper, "are:")

for num in range(lower, upper + 1):

if num > 1:

for i in range(2, num):

if (num % i) == 0:

break

else:

print(num,end=" ")

**Result:**

Prime numbers between 900 and 1000 are:

907 911 919 929 937 941 947 953 967 971 977 983 991 997

**Example:**

# Python program to find the factorial of a number provided by the user.

num = int(input("Enter a number: "))

factorial = 1

if num < 0:

print("Sorry, factorial does not exist for negative numbers")

elif num == 0:

print("The factorial of 0 is 1")

else:

for i in range(1,num + 1):

factorial = factorial\*i

print("The factorial of",num,"is",factorial)

**Result:**

Enter a number: 5

The factorial of 5 is 120

**Example:**

# Multiplication table from 1 to 10 of a number

num = int(input("Display multiplication table of? "))

for i in range(1, 11):

print(num, 'x', i, '=', num\*i)

**Result:**

Display multiplication table of? 9

9 x 1 = 9

9 x 2 = 18

9 x 3 = 27

9 x 4 = 36

9 x 5 = 45

9 x 6 = 54

9 x 7 = 63

9 x 8 = 72

9 x 9 = 81

9 x 10 = 90

**Example:**

# A Fibonacci sequence is the integer sequence of 0, 1, 1, 2, 3, 5, 8....

# Program to display the Fibonacci sequence up to n-th term

nterms = int(input("How many terms? "))

# first two terms

n1, n2 = 0, 1

count = 0

# check if the number of terms is valid

if nterms <= 0:

print("Please enter a positive integer")

elif nterms == 1:

print("Fibonacci sequence upto",nterms,":")

print(n1)

else:

print("Fibonacci sequence:")

while count < nterms:

print(n1)

nth = n1 + n2

# update values

n1 = n2

n2 = nth

count += 1

**Result:**

How many terms? 6

Fibonacci sequence:

0

1

1

2

3

5

**Example:**

num = int(input("Enter a number to check whether it is an armstrong number: "))

order = len(str(num))

# initialize sum

sum = 0

# find the sum of the cube of each digit

temp = num

while temp > 0:

digit = temp % 10

sum += digit \*\* order

temp //= 10

# display the result

if num == sum:

print(num,"is an Armstrong number")

else:

print(num,"is not an Armstrong number")

**Result:**

Enter a number to check whether it is an armstrong number: 54748

54748 is an Armstrong number

**Example:**

# Program to check Armstrong numbers in a certain interval

lower = 100

upper = 2000

for num in range(lower, upper + 1):

# order of number

order = len(str(num))

# initialize sum

sum = 0

temp = num

while temp > 0:

digit = temp % 10

sum += digit \*\* order

temp //= 10

if num == sum:

print(num)

**Result:**

153

370

371

407

1634

**Example:**

# Sum of natural numbers up to num

num = 4

if num < 0:

print("Enter a positive number")

else:

sum = 0

# use while loop to iterate until zero

while(num > 0):

sum += num

num -= 1

print("The sum is", sum)

**Result:**

The sum is 10

//////////////////////////////

The L.C.M. can only be greater than or equal to the largest number.

///////////////////////////

**Example:**

# Python Program to find the L.C.M. of two numbers

x = 30

y = 24

if x > y:

greater = x

else:

greater = y

while(True):

if((greater % x == 0) and (greater % y == 0)):

lcm = greater

break

greater += 1

print("The L.C.M. is", lcm)

**Result:**

The L.C.M. is 120

/////////////////////////////////

x\*y = LCM \* GCD

/////////////////////////////////

**Example:**

#Program to find LCM using GCD

x = 54

x1 = x

y = 24

y1 = y

while(y):

x, y = y, x % y

lcm = (x1\*y1)//x

print(lcm)

**Result:**

216

/////////////////////////////////

The standard form of a quadratic equation is:

ax2 + bx + c = 0, where

a, b and c are real numbers and

a ≠ 0

////////////////////////////////////

**Example:**

# Solve the quadratic equation ax\*\*2 + bx + c = 0

# import complex math module

import cmath

a = 1

b = 5

c = 6

# calculate the discriminant

d = (b\*\*2) - (4\*a\*c)

# find two solutions

sol1 = (-b-cmath.sqrt(d))/(2\*a)

sol2 = (-b+cmath.sqrt(d))/(2\*a)

print('The solutions are',sol1,'and', sol2)

**Result:**

The solutions are (-3+0j) and (-2+0j)

**Assignments**

1

Pattern printing,

2

Identify negative, positive, even, odd numbers

3

Identify hcf using loop

4

Assignment on different types of variable swapping.

5

Identify leap year with nested if.

6

Identify prime numbers, factorial, Fibonacci sequence, Armstrong numbers, LCM

7

Solution using quadratic equation