



# Python Flow Control

if...else

for Loop

while loop

Break and continue

Pass statement

## if...else

- 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.

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

- 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.

# If the number is positive, we print an appropriate message

```
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.")
```

## Exercise – if

- Write a program to give a discount of 10% if the total bill amount exceeds 1000.

#program to give discount if the total amount is greater than 1000.

```
shoppingAmount = int(input("Enter the shopping Amount : "))
```

```
if(shoppingAmount > 1000):
```

```
    discount = 10 / 100 * shoppingAmount
```

```
    print('Discount = ' , discount)
```

```
    shoppingAmount -= discount
```

```
print('Final Shopping Amount = ' , shoppingAmount)
```

# if...else Statement

- 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.

## Syntax of if...else

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

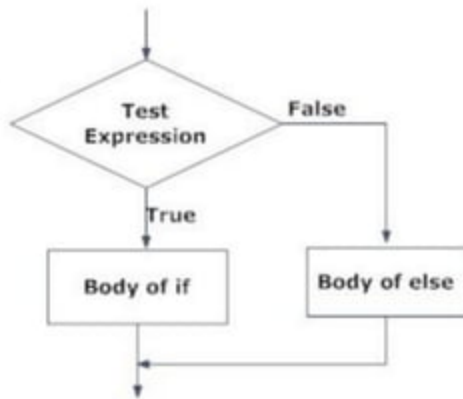


Fig: Operation of if...else statement

```
# Program checks if the number is positive or negative
# And displays an appropriate message
```

```
num = 3
```

```
# Try these two variations as well.
```

```
# num = -5
```

```
# num = 0
```

```
if num >= 0:
```

```
    print("Positive or Zero")
```

```
else:
```

```
    print("Negative number")
```

## Exercise

- Write a program to check if a given number is a multiple of 5.



# if...elif...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 elifblocks.

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

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

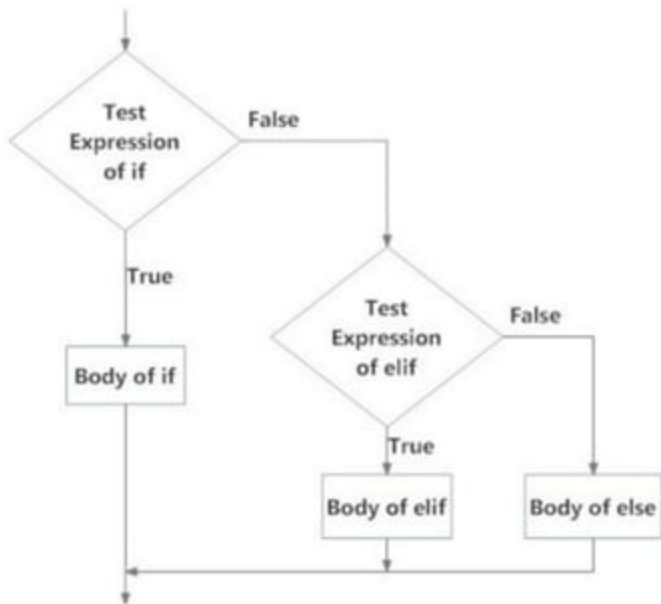


Fig: Operation of if...elif...else statement

```
if num > 0:
    print("Positive")
else:
    if num < 0:
        print("Negative")
    else:
        print("Zero")
```

```
if num > 0:
    print("Positive")
elif num < 0:
    print("Negative")
else:
    print("Zero")
```

## Programming Task

- Minimum age to Cast your Vote : 18
- Minimum age to Contest an Election: 25
  
- Given the age verify if the person can vote and can s(he) contest an election.

# Exercise

- Write a program to check if a given year is leap year or not.
- Logic:
  - if a year is not divisible by 4, its not a leap year.
  - If a year is divisible by 4 and not divisible by 100, it's a leap year.
  - If a year is divisible by 4 and 100 then it should be divisible by 400 to be a leap year

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

```
year = int(input("Enter a year: "))
```

```
if (year % 4) == 0:
```

```
    if (year % 100) == 0:
```

```
        if (year % 400) == 0:
```

```
            print(year , "is a leap year")
```

```
        else:
```

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

```
    else:
```

```
        print(year, "is a leap year")
```

```
else:
```

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

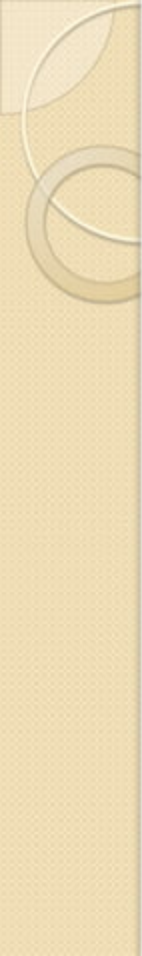
```
|
```

```
# Python program to check if the input year is a leap year or not
year = int(input("Enter a year: "))

if (year % 4) == 0 and (year % 100) != 0:
    print(year , "is a leap year")
else:
    if ((year % 100) == 0) and ((year % 400) == 0):
        print(year , "is a leap year")
    else:
        print(year , "is not a leap year")
```

```
#program to check if a given year is leap year or not.
year = int(input("Enter a year: "))

if ((year % 4) == 0 and (year % 100) != 0) or ((year % 100) == 0 and (year % 400)
== 0):
    print(year , "is a leap year")
else:
    print(year , "is not a leap year")
```





# Loops

- **Loops are used in programming to repeat a specific block of code.**
- **Looping Constructs in Python**
  - **while**
  - **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 beforehand, the number of times to iterate.

## Syntax of while Loop

```
while test_expression:  
    Body of while
```

```
#program to calculate the gcd of two numbers
```

```
a = int(input("Enter first number"))
```

```
b = int(input("Enter second number"))
```

```
while(a != b):
```

```
    if(a > b):
```

```
        a -= b
```

```
    else:
```

```
        b -= a
```

```
print("Gcd = " , a)
```

```
while(b != 0):
```

```
    temp = a % b
```

```
    a = b
```

```
    b = temp
```

```
print("Gcd = " , a)
```

- Write a program to print the number of digits of a given number using while loop.

```
num = int(input("Enter a number "))  
digits = 0
```

```
while num != 0:  
    num = num // 10;  
    print ("Num = " , num)  
    digits = digits + 1
```

```
print ("Number of digits = " , digits)
```

```
print("Program to verify if a given number is prime ")

n = int(input("Enter the number ") )

div = 2
flag = True
while div < n:
    if n % div == 0:
        flag = False
        div = div + 1

if flag == True:
    print("Prime")
else:
    print("Not Prime")
```

# For loop

- For loops iterate over a given sequence.

## 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.

```
# Program to find the sum of all numbers stored in a list

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

# variable to store the sum
totalSum = 0

# iterate over the list
for val in numbers:
    totalSum = totalSum + val

print("The sum is", totalSum)
```

# Range Function

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

```
# Output: range(0, 10)
print(range(10))
```

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

```
# Output: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(list(range(10)))
```



## Range function

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

```
# Output: [2, 3, 4, 5, 6, 7]  
print(list(range(2, 8)))
```

```
# Output: [2, 5, 8, 11, 14, 17]  
print(list(range(2, 20, 3)))
```

```
print("Program to verify if a given number is prime ")
n = int(input("Enter the number "))

flag = True

for div in range(2,n):
    if n % div == 0:
        flag = False

print("Prime" if flag == True else "Not Prime")
```

## Exercise


- Write a program to calculate the factorial of a given number.

```
#program to calculate the factorial of a number.  
n = int(input('Enter a number :'))  
res = 1  
  
for x in range(1,n + 1):  
    res = res * x  
  
print('Factorial of ' , n , ' = ' , res)
```

## break and continue


- 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.
- **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 var in sequence:  
    # codes inside for loop  
    if condition:  
        break  
    # codes inside for loop  
# codes outside for loop
```



---

```
while test expression:  
    # codes inside while loop  
    if condition:  
        break  
    # codes inside while loop  
# codes outside while loop
```



## Exercise

- Write a program to check if a given number is prime or not.
- If a number is divisible by any number between 2 and  $n-1$ , its not prime, otherwise prime

```
print("Program to verify if a given number is prime ")
n = int(input("Enter the number ") )

flag = True

for div in range(2,n):
    if n % div == 0:
        flag = False
        break

print("div = " ,div)
print("Prime" if flag == True else "Not Prime")
```



# 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.

```
for var in sequence:  
    # codes inside for loop  
    if condition:  
        continue  
    # codes inside for loop  
  
# codes outside for loop
```

---

```
while test expression:  
    # codes inside while loop  
    if condition:  
        continue  
    # codes inside while loop  
  
# codes outside while loop
```

```
for val in range(20):  
    if val % 3 == 0:  
        continue  
    print(val)  
  
print("The end")
```

# for loop with else

- A for loop can have an optional else block as well. The else part is executed if the items in the sequence used in for loop exhausts.
- break statement can be used to stop a for loop. In such case, the else part is ignored.
- Hence, a for loop's else part runs if no break occurs.

```
#program to demonstrate for with else
digits = [0, 1, 5]

for i in digits:
    print(i)
else:
    print("No items left.")
```

# for loop with else

```
print("Program to verify if a given number is prime ")  
  
n = int(input("Enter the number ") )  
  
for div in range(2,n):  
    if n % div == 0:  
        print("Not Prime..divisible by",div)  
        break  
else:  
    print("Prime")
```

# Nested Loop

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

```
lower = int(input("Enter lower range: "))
```

```
upper = int(input("Enter upper range: "))
```

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

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

```
    # prime numbers are greater than 1
```

```
    if num > 1:
```

```
        for i in range(2, num):
```

```
            if (num % i) == 0:
```

```
                break
```

```
        else:
```

```
            print(num)
```

```
|
```

# while loop with else

- Same as that of for loop, we can have an optional else block with while loop as well.

```
n = 9
i = 2
while i < n:
    if(n % i == 0):
        print('Not Prime...Divisible by ' , i)
        break
    i = i + 1
else:
    print('Number is prime')
|
```

# 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 an 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
```