Definition and Classification

24 June 2024 02:12 PM

Flow Control Statements:

1. In any language, the flow of execution of statements follow Water falls Approach (Top to Bottom)
2. Flow control statements are the special statements which are used for controlling or changing the actual flow of execution.

Classification of flow control statements:

1. Conditional or Decisional statements
2. Looping statements

Conditional or Decisional Statements:

In this type, based on one condition, we will whether to execute set of instructions or not to execute set of instructions.

In python, we have 4 types of conditional statements

1. If condition
2. If elif condition
3. If -else condition
4. Nested if condition.

Indentation: (25-June)

Indentation is a common space which is used to represent a block in python.

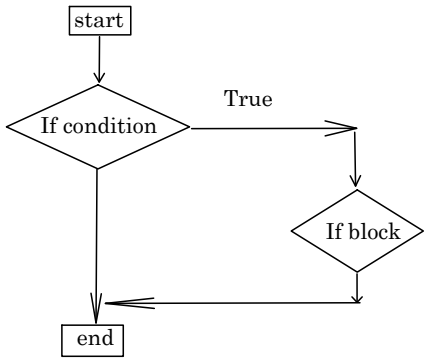
If condition:

1. When we have only one condition to check, then we use if condition.

syntax:

```
if condition:
    statements of if block
```

Flow:-



Example: Write a program to print the given number if it is greater than 100

```
a = int(input('Enter a int value: '))
if a>100:
    print(a)
```

Example: Write a program to print the given number if it is even number.

```
# If condition
a = int(input('Enter the integer value: '))
if a%2 == 0:
    print(a)
```

If - else condition:

1. When we have 2 conditions to check, we will use if-else condition.
2. Among 2 conditions, we will have
   a. Mandatory condition
   b. Default condition

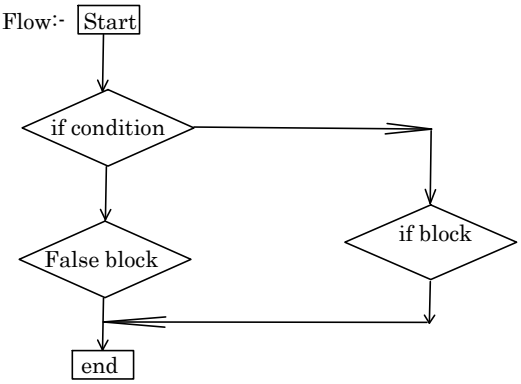
Note:

We can choose any condition as mandatory and the remaining as condition as default condition.

Syntax:

```
If condition:
    If bock statements
Else:
    Else block statements
```

Flow:-



Example: Write a program to check the given number is even or odd.

```
# If-Else condition
# even as mandatory condition
a = int(input('Enter the integer value: '))
if a%2 ==0:
    print('The given number is odd')
else:
    print('The given number is odd')
```

```
# odd as mandatory condition
a = int(input('Enter the integer value: '))
if a%2 ==1:
    print(a,' is odd')
else:
    print(f'{a} is even')
```

Write a program to print the given string is palindrome or not.

```
string = input('Enter a string: ')
if string == string[::-1]:
    print(f'{string} is a palindrome')
else:
    print(f'{string} is not a palindrome')
```

Write a program to find the maximum among given two numbers.

```
a = int(input('enter first number: '))
b = int(input('enter second number: '))
if a>b:
    print(a)
else:
    print(b)
```

Write a program to check given words are anagrams are not.
Note: Anagram strings are the strings which has same set of characters in both strings. Example: Art and Rat, Listen and Sile nt.

```
a = input('Enter first string: ')
b = input('Enter second string: ')
if list(a).sort() ==list(b).sort():
    print(f'{a.upper()} and {b.upper()} are anagrams')
else:
    print(f'{a.upper()} and {b.upper()} and are not anagrams')
```

```
a = input('Enter a string: ')
b = input('Enter second string: ')
if sorted(a) == sorted(b):
    print(f'{a.upper()} and {b.upper()} are anagrams')
else:
    print(f'{a.upper()} and{b.upper()} are not anagrams')
```

```
Enter a string: silent
Enter second string: listen
SILENT and LISTEN are anagrams
```

Write a program to print the given number if the given number is even number and if it is greater than 100.

```
n = int(input('Enter an integer value: '))
if n%2 ==0 and n>100:
    print(n)
```

```
Enter an integer value: 102
102
```

Write a program to print the given year is leap year or not.

```
year = int(input())
if (year%100 ==0 and year%400 == 0) or (year%100!=0 and year%4 == 0):
    print(f'{year} is a leap year')
else:
    print(f'{year} is not a leap year')
```

If elif conditional statements

26 June 2024 05:24 PM

If elif condition:

When we have more than 2 conditions to check, then we will use if-elif condition

Syntax:

If condition:

    if block

Elif condition:

    Elif block

Elif condition:

    Elif block

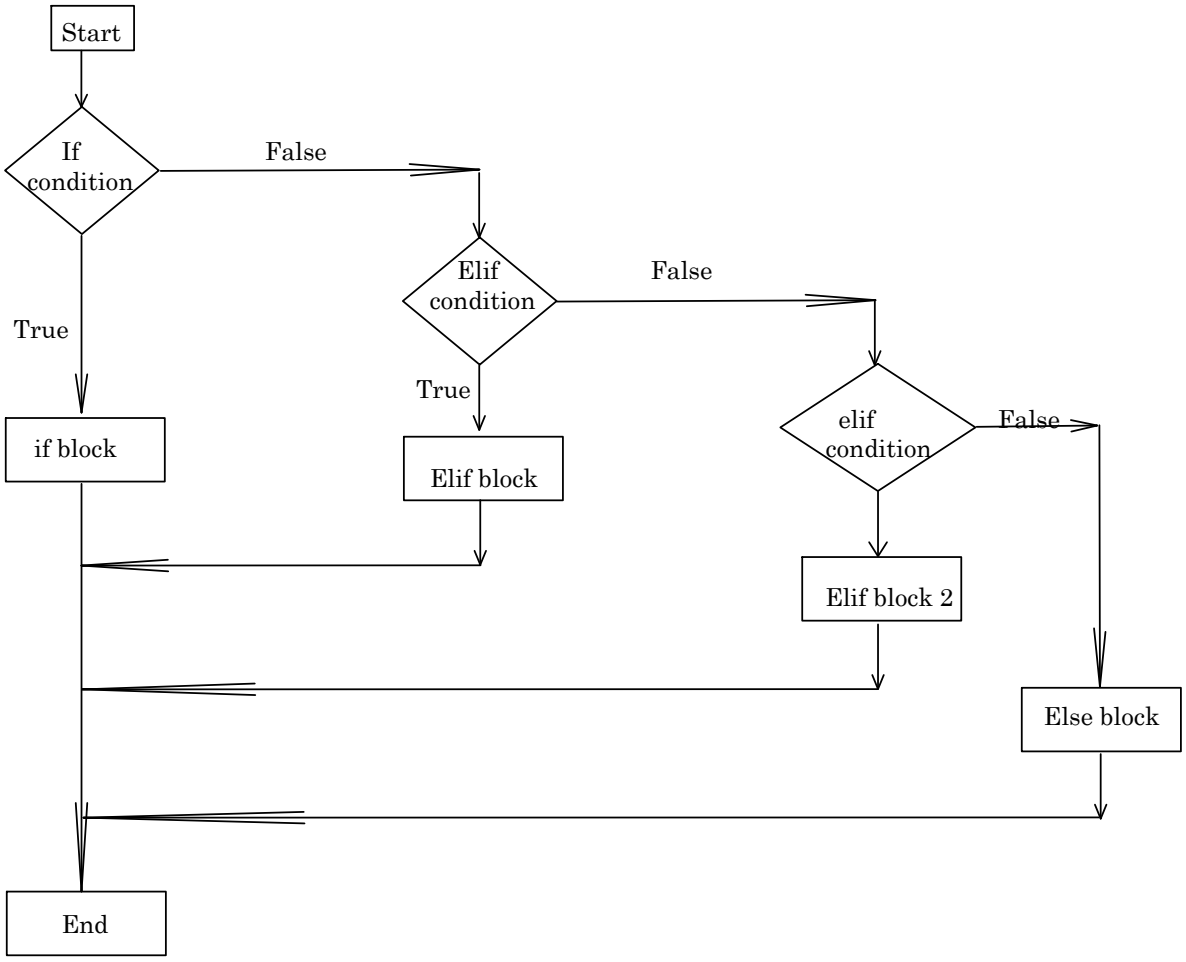
Elif condition:

    Nth elif block

Else:

    Else block

Flow:



Example:

```
print('''Menu
1. Indian
2. Russian
3. Italian''')

choice = int(input('please enter your choice: '))
if choice == 1:
    print('Serve Indian food')
elif choice ==2:
    print('Serve Russian food')
elif choice ==3:
    print('Serve Italian food')
else:
    print('Sorry, Please visit other restaurant')
```

Write a program to find out the relationship between given 2 numbers.

```
n1 = int(input('Enter a number: '))
n2 = int(input('Enter second number: '))
if n1==n2:
    print(f'{n1} is equal to {n2}')
elif n1>n2:
    print(f'{n1} is greater than {n2}')
else:
    print(f'{n1} is less than {n2}')
```

WAPT check the relationship between 3 numbers.

```
# To find max between 3 numbers
a,b,c = 2,3,8
if a>b and c>c:
    print('a is max')
elif b>c:
    print('b is max')
else:
    print('c is max')
```

```
c is max
```

```
# To find realationship between 3 numbers
a,b,c = 2,3,8
if a==b and a==c:
    print('all are equal')
elif a>b and c>c:
    print('a is max')
elif b>c:
    print('b is max')
else:
    print('c is max')
```

```
c is max
```

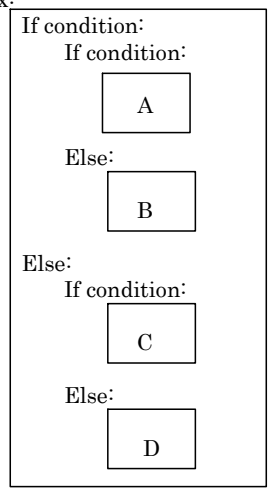
Nested if

27 June 2024 04:25 PM

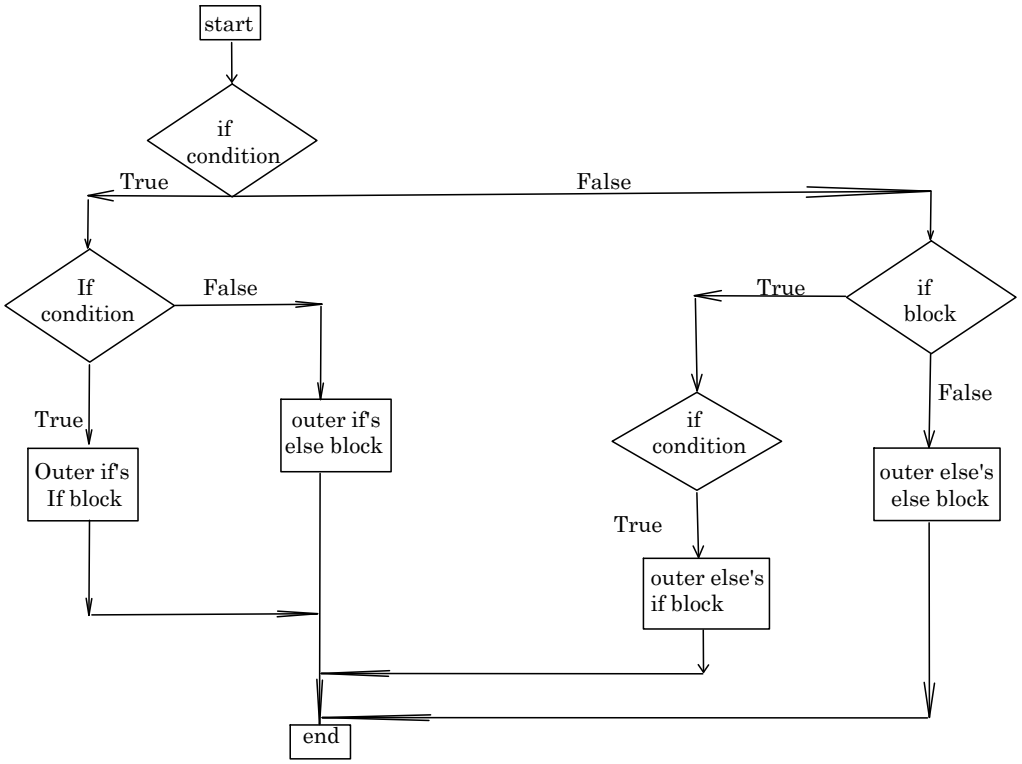
Nested if :

- It is the process of creating the condition inside another condition
- When we have more than 2 conditions to check, then we will use nested if condition.

Syntax:



Flow of execution:



Note:

- IF-ELIFs: BEST for COMPLEX CODING TASKS (multiple conditions) because it will REDUCE the LINES OF CODE. But will make the program RUN SLOWER since the control will have to check for entire conditions in the if statement multiple times unlike the n ested if statements where the control enters the block one by one, only if the conditions are satisfied. Which is CODE REDUNDANCY.

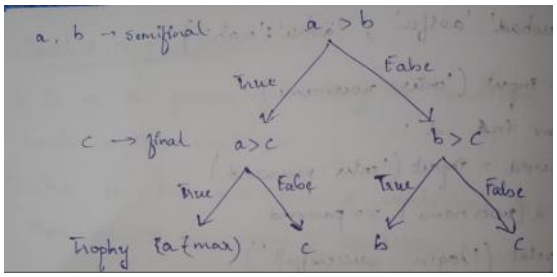
- NESTED Ifs: FASTER for the same reason that there is no such code redundancy. The control will enter a block of code only if the condition is True. BEST for TIME COMPLEXITY ISSUES.

Example:

```
d=  {'harshad' : 'asifa', 'ashu' : 'mahi' }
username = input('Enter username: ')
if username in d:
    password = input('Enter password: ')
    if d[username] ==password:
        print('login successfull')
    else:
        print('passowrd not matching')
else:
    print('Given username is not present in the Database')
```

```
Enter username: harshad
Enter password: asifa
login successfull
```

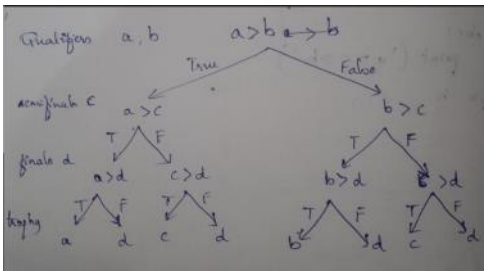
WAPT print maximum among given three numbers using nested if.




```
a,b,c = 5,10,15
if a>b:
    if a>c:
        print('a is max')
    else:
        print('c is max')
else:
    if b>c:
        print('b is max')
    else:
        print('c is max')
```

```
c is max
```

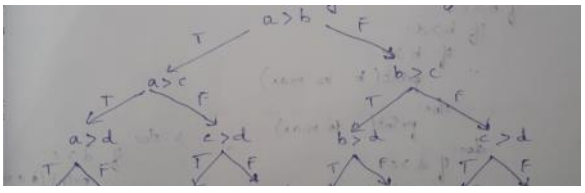
WAPT print maximum among given 4 numbers using nested if.

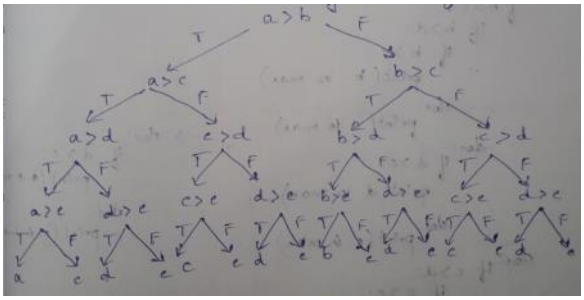



```
a,b,c,d = 5,10,15,2
if a>b:
    if a>c:
        if a>d:
            print('a is max')
        else:
            print('d is max')
    else:
        if c>d:
            print('c is max')
        else:
            print('d is max')
else:
    if b>c:
        if b>d:
            print('b is max')
        else:
            print('d is max')
    else:
        if c>d:
            print('c is max')
        else:
            print('d is max')
```

```
c is max
```

WAPT print maximum among given 5 numbers using nested if.






```
if a>b:
    if a>c:
        if a>d:
            if a>e:
                print('a is max')
            else:
                print('e is max')
        else:
            if d>e:
                print('d is max')
            else:
                print('e is max')
    else:
        if c>d:
            if c>e:
                print('c is max')
            else:
                print('e is max')
        else:
            if d>e:
                print('d is max')
            else:
                print('e is max')
else:
    if b>c:
        if b>d:
            if b>e:
                print('b is max')
            else:
                print('e is max')
        else:
            if d>e:
                print('d is max')
            else:
                print('e is max')
    else:
        if c>d:
            if c>e:
                print('c is max')
            else:
                print('e is max')
        else:
            if d>e:
                print('d is max')
            else:
                print('e is max')
```

Looping Statements

28 June 2024 04:58 PM

When we have to execute some set of instructions repeatedly, then we use looping statements.
In python, we have 2 types of looping statements:

1. For loop (when we know the number of iterations to be done)
2. While loop (when we are not sure about the number of iterations)

For loop:

For loop will perform below mentioned operations

- Initialization (initialize a variable and assign a value to that variable)
- Traversing (travel to next value available for looping)

We can use for loop in 3 ways:

1. With Collection data types
2. With **range** function (integers)
3. With **range** function and CDT (Index Positions)

Using For loop with the Collection Data Type:

- Whenever we want to perform some operations on the elements of the collection data type directly with elements, then we will use for loop with collection.
  Syntax of for loop with collection:

```
for variable in CDT:
    Statements of
    for loop with CDT
```

```python
# for loop with string
s= 'bye'
for i in s:
    print('I am in for block')
    print(i)

# for loop with list
l = ['bye', 878,45,90]
for i in l:
    print("I'm in for block")
    print(i)

# for loop with tuple
t = ('bye', 878,45,[90,78])
for i in t:
    print("I'm in for block")
    print(i)
```

```python
#for loop with set:
s = {'bye', 878,45,(90,78)}
for i in s:
    print("Im in forblock")
    print(i)

# for loop with dictionary (directly keys are taken)
d = {'name':'ashu', 'age':3}
for key in d:
    print(key, key[d])

# for loop with dicyionary's items method
d = {'name':'ashu', 'age':3}
for k,v in d.items():
    print(k,v)
```

For Loop with CDTs

01 July 2024 04:18 PM

Note:
- While using for loop with CDT, note that the control will not even enter the loop if the CDT is empty; say in case of an empty string.

WAPT check how many elements are present in given CDT.

```
cdt = eval(input('Enter a collection data type: '))
count = 0        # print(len(cdt)) --> using the built-in function
for i in cdt:    # prefer to show the logic instead of using easy built-in functions
    count += 1
print(count)
```

Steps:
- Take CDT as input and use eval function since the data type of the given collection is not known.
- Assign count = 0 for 2 reasons:
    - Initiate the variable count
    - Suppose if the given CDT is empty, always begin with count = 0.
- Begin the for loop by fetching the first element of the given CDT and assigning it to a variable and increase the count by 1.
- Check for 2nd element, and continue the process till all the elements in the CDT are checked for.
- When all the iterations are done, the control will exit from the loop and then print the count of elements at the end.

WAPT print how many times the given substring is present in the given string.

```
str_ = input('Enter a string: ')
sub = input('Enter a character : ')
count = 0
for i in str_:
    if i == sub:
        count += 1
print(count)
```

```
Enter a string: hai123ii
Enter a character : i
3
```

Steps:
- Enter the main string and the substring to be checked (hai123ii)
- Assign count = 0 (if the given substring is not present in the given string) (substring: i)
- In the for loop, fetch the first element and check if it same as the given substring. (h == i:)
- If the condition is True, increment the count, and go for the next element of the CDT (count = 1)
- If the condition is false, take the next element and check for the condition and continue the process till the last element. (count = 0)
- After all the iterations are done, exit from the for loop and print the count at the end.

WAPT print how many numbers are present in the given string.

```
str_ = input('Enter a string: ')
count = 0
for i in str_:
    if i.isdigit():
#     if i in '0123456789': or use ord function to check the ASCII value
        count += 1
print(count)
```

```
Enter a string: hai21345bye
5
```

WAPT find the sum of digits in the given number. (02-July)

```
str_ = input('Enter a string: ')
sum_ = 0
for i in s:
    if i.isdigit():
        sum_ +=int(i)
print(sum_)
```

```
Enter a string: hai8974bye
25
```

Steps:
- Take an input string
- Consider the sum of integers sum_ = 0 supposing that there are no integers in the given string.
- Beginning the for loop:
    - Consider the first element of the string and assign a variable 'i' to the first element.
    - Check if the variable is digit or not
    - If the variable is a digit, convert the string of integer to integer and add to the sum of numbers sum_

WAPT find the sum of even digits present in a given string

```
str_ = input('Enter a string: ')
sum_ = 0
for i in str_:
    if i.isdigit() and int(i)%2 ==0:
        sum_ += int(i)
print(sum_)
```

```
Enter a string: kjehfi7983nof798
16
```

Or

```
str_ = input('Enter a string: ')
sum_ = 0
for i in str_:
    if i.isdigit():
        if int(i)%2 ==0:
            sum_ += int(i)
print(sum_)
```

Note:

- The condition to check for the integer should be mentioned first, so that the control will go for next condition only if it is an integer.
- Reversing the condition will throw an error.
- Nested if is better in this case because checking for even or odd will be done only if it is an integer.

WAPT count how many even digits and odd digits are present in a given string

```
str_ = input('Enter a string: ')
even_count = 0
odd_count = 0
for i in str_:
    if i.isdigit():
        if int(i)%2 ==0:
            even_count += 1
        else:
            odd_count += 1
print('Sum of even numbers: ', even_count)
print('Sum of odd numbers: ', odd_count)
```

```
Enter a string: hai68737534bye
Sum of even numbers:  3
Sum of odd numbers:  5
```

WAPT find sum of even and odd digits present in a given string.

```
str_ = input('Enter a string: ')
even_sum = 0
odd_sum = 0
for i in str_:
    if i.isdigit():
        if int(i)%2 == 0:
            even_sum += int(i)
        else:
            odd_sum += int(i)
print('Sum of even numbers: ', even_sum)
print('sum of odd numbers: ', odd_sum)
```

```
Enter a string: lijug67863mhf
Sum of even numbers:  20
sum of odd numbers:  10
```

WAPT find the absolute difference between sum of even digits and odd digits.

```
str_ = input('Enter a string: ')
even_sum = 0
odd_sum = 0
for i in str_:
    if i.isdigit():
        if int(i)%2 == 0:
            even_sum += int(i)
        else:
            odd_sum += int(i)
if even_sum > odd_sum:
    sum_ = even_sum-odd_sum
else:
    sum_ = odd_sum-even_sum
print('The absolute difference between the sum is : ', sum_)
```

WAPT print how many vowels are present in given string.

```
str_ = input('Enter a string: ')
count = 0
for i in str_:
    if i.lower() in ['a', 'e', 'i', 'o', 'u']:
        count += 1
#     if i in 'aeiouAEIOU':
#         count += 1
print(count)
```

```
Enter a string: ndkyinekaenk76849ji
5
```

WAPT count how many consonants are present in given string.

```
str_ = input('Enter a string: ')
```

```
str_  = input('Enter a string: ')
count = 0
for i in str_:
#     if i.isalpha() and i not in 'AEIOUaeiou':
    if i.isalpha():
        if i not in 'aeiouAEIOU':
            count += 1
print(count)
```

```
Enter a string: hai123
1
```

WAPT print how many alphanumerical values are present in given string.

```
str_  = input('Enter a string: ')
count = 0
for i in str_:
#     if i.isalpha() or i.isdigit():
    if i.isalnum():
        count += 1
print(count)
```

```
Enter a string: hai123@$1bye
10
```

WAPT print how many special characters are present in given string.

```
str_  = input('Enter a string: ')
count = 0
for i in str_:
#     if i.isalnum() == 0:
    if not i.isalnum():
        count +=1
print(count)
```

```
Enter a string: haiHAI123@#$uU
3
```

WAPT find the sum of digits present in the given list. 03-July

```
# WAPT find the sum of digits present in the given list
list_ = eval(input('Enter a list: '))
sum_ = 0
for i in list_:
    if type(i) == int: # or isinstance(i, int)
        sum_ += i
print(sum_)
```

```
Enter a list: 1,1.5,1000, 'j'
1002.5
```

Note:

Isinstance is a function which is used for checking the given data is an object of specified data type. If the data is an object of specified data, it returns True, else it returns False.

Syntax:
Isinstance(data, data-type)

WAPT find the maximum number in a given list.

```
list_ = eval(input('Enter a list: '))
max_ = list_[0]
for i in list_:
    if i>max_:
        max_ = i
print(max_)
```

```
Enter a list: 1,2,5,8,6,3
8
```

WAPT print how many times each and every element is present in given collection.

```
cdt = eval(input('Enter a CDT: '))
count = {}
for i in var:
    count[i] = var.count(i)
print(count)
```

```
Enter a CDT: 'haaii'
{'h': 1, 'a': 2, 'i': 2}
```

```
cdt = eval(input('Enter a CDT: '))
count = {}
for i in cdt:
    if i not in count.keys():
        count[i] = 1
    else:
        count[i] += 1
print(count)
```

```
Enter a CDT: 'haaii'
{'h': 1, 'a': 2, 'i': 2}
```

```
cdt = eval(input('Enter a CDT: '))
count = {}.fromkeys(cdt, 0)
for i in cdt:
    count[i] += 1
print(count)
```

```
Enter a CDT: 'haaii'
{'h': 1, 'a': 2, 'i': 2}
```

```
str_ = 'haaii'
set_ = set(str_)
for i in set_:
    print(i, str_.count(i), end = '    ', sep='-->')
```

```
a-->2    h-->1    i-->2
```

```
d = {}
for i in str_:
    d[i] = str_.count(i)
print(d)
```

```
{'h': 1, 'a': 2, 'i': 2}
```

WAPT print most repeated character in a given string. (get first if the result is more than one)

```
s = input('Enter a string: ')
mrc = ''
c=0
for i in s:
    if s.count(i)>c:
        mrc = i
        c=s.count(i)
print(mrc)
```

```
Enter a string: harshad
h
```

WAPT most repeated word in a given string. 04-July

```
s = input()
words = s.split()
mrw = ''
c=0
for word in words:
    if words.count(word) > c:
        c = words.count(word)
        mrw = word
print(mrw)
```

```
hi python hello python
python
```

WAPT lengthiest word in a given string.

```
words = input().split()
count = 0
l_word = ''
for word in words:
    if len(word)>count:
        count = len(word)
        l_word = word
print(l_word)
```

```
hi whats up
whats
```

Or

```
print(max(words, key = len))
```

```
python
```

By default, max will provide the value with maximum ascii value as output. Hence using key as 'len' will check the maximum using len function.

For loop with Range

04 July 2024 04:48 PM

Range function:

Range function is used for providing LIMITS for the execution of FOR LOOP.

Syntax:

Range(starting_limit, ending_limit+-1, updation)

Note:

- Default value of updation is 1
- If we provide only one value for range, it will consider it as the ending limit and starts from zero.

Example:

```
70->89 --> range(70,90)
78->60 --> range(78,59,-1)
-1 -> -100 --> range(-1, -101, -1)
-121 -> -100 --> range(-121, -99)
```

Direction	Updation	Ending value
Left --> Right	+1	Larger number than given ending value
Right --> Left	-1	Smaller number than given ending value

For loop with Range:

For loop with range is used whenever we want to perform some operations based on number limits.

Syntax:

For varible in range(sl, el+-1, updation):

Statements of for loop

Example:

```
for i in range(1,4):
    print(i)
```

```
1
2
3
```

WAPT print squares of first 10 positive numbers.

```
for i in range(10):
    print(i**2, end = ' ')
```

```
0 1 4 9 16 25 36 49 64 81
```

WAPT print first 10 even numbers.

```
for i in range(19):
    if i%2 == 0:
        print(i, end = ' ')
```

```
0 2 4 6 8 10 12 14 16 18
```

```
for i in range(0,19,2):
    print(i, end = ' ')
```

```
0 2 4 6 8 10 12 14 16 18
```

WAPT print even numbers in a given range.

```python
#  even numbers in given range
low_limit = int(input('Enter lower limit of range: '))
up_limit = int(input('enter the upper limit for the range: '))
for i in range(low_limit, up_limit+1):
    if i%2==0:
        print(i, end = ' ')
```

```
Enter lower limit of range: 5
enter the upper limit for the range: 16
6 8 10 12 14 16
```

WAPT find sum of numbers in a given range.

```python
sl = int(input('enter the lower limit of range: '))
el = int(input('enter the upper limit of the range: '))
sum_ = 0
for i in range(sl, el+1):
    sum_ += i
print(sum_)
```

```
enter the lower limit of range: 5
enter the upper limit of the range: 25
315
```

WAPT find the sum of first n numbers.

```python
n = int(input('enter the upper limit: '))
sum_ = 0
for i in range(1,n+1):
    sum_ += i
print(sum_)
```

```
enter the upper limit: 6
21
```

WAPT find the factorial of a given number.

```python
n = int(input('enter the a number: '))
factorial = 1   # 0! = 1
for i in range(1,n+1):
    factorial*= i
print(factorial)
```

```
enter the a number: 6
720
```

WAPT find given number is perfect number or not.

Note: If the given number is equal to sum of its divisors, then we call the given number is perfect number.

```python
sum_ = 0
n = int(input('enter a number: '))
for i in range(1,n//2+1):
    if n%i ==0:
        sum_ +=i
if sum_ == n:
    print(f'{n} number is a perfect number')
else:
    print(f'{n} number is not a perfect number')
```

```
enter a number: 28
28 number is a perfect number
```

For loop with Range and CDT

06 July 2024 09:55 AM

For loop with range and CDT:

- It is used whenever we want to perform some operations on the elements based on index positions. (indexing)
- It is also used whenever we want to deal with multiple elements. (slicing)

[Note:

Use for loop with CDT when 1) we want each and every element from given collection
2) extract 1 element directly

Use for loop with range when 1) we want each and every number from given collection

Use for loop with range and CDT when 1) we want to perform operations based on index positions
2) we want to extract multiple elements]

Syntax:

```
for variable in range(len(CDT)):
    statements of for loop
    with range and CDT
```

Example:

```
s = 'hello'
for ip in range(len(s)):
    print(ip, s[ip])
```

```
0 h
1 e
2 l
3 l
4 o
```

WAPT print all the index positions where h is present in a given string.

```
s = input('Enter a string: ')
for ip in range(len(s)):
    if s[ip] in'hH':
        print(ip)
```

```
hai python
0
7
```

WAPT print the elements which are in even index positions of given string.

```
s = input()
for ip in range(0,len(s),2):
        print(s[ip])

for ip in range(len(s)):
    if ip%2 ==0:
        print(s[ip])
```

```
hai python
h
i
p
t
o
```

WAPT print the index positions of vowels.

```
s=input()
for ip in range(len(s)):
    if s[ip] in 'aeiouAEIOU':
        print(ip)
```

```
hai python
1
2
8
```

WAPT replace all the vowels with their index positions.

```
s = input()
new=''
for ip in range(len(s)):
    if s[ip] in 'aeiouAEIOU':
        new += str(ip)
    else:
        new += s[ip]
print(new)
```

```
hai python
h12 pyth8n
```

WAPT odd numbers present in odd index positions of a given string.

```
s = input()
for ip in range(len(s)):
    if s[ip].isdigit():
        if int(s[ip])%2 == 1:
            if ip%2 == 1:
                print(s[ip], end = ' ')
```

```
h123456789
1 3 5 7 9
```

WAPT odd digits present in even index positions in the given string.

```
s = input()
for ip in range(len(s)):
    if s[ip].isdigit():
        if int(s[ip])%2==1 and ip%2==0:
            print(s[ip], end = ' ')
```

```
h 123456789
1 3 5 7 9
```

WAPT print the index positions of even digits present in even index positions.

```
s = input()
for ip in range(len(s)):
    if s[ip].isdigit():
        if int(s[ip])%2==0 and ip%2==0:
            print(ip, end = ' ')
```

```
h987456321
2 4 6 8
```

```
even = '02468'
s = input()
for ip in range(0, len(s),2):
    if s[ip] in even:
        print(s[ip], end = ' ')
```

```
h987654321
8 6 4 2
```

Find the sum of even digits present in odd index positions in the given string.

```
s = input()
sum_ = 0
for ip in range(len(s)):
    if s[ip].isdigit():
        if int(s[ip])%2==0 and ip%2==1 :
            sum_+=int(s[ip])
print(f'Sum is {sum_}')
```

```
h 123456789
Sum is 20
```

WAPT reverse a given string without using slicing.

```
s = input()
rs = ''
for i in s:
    rs = i + rs
print(rs)

# can also use range(-1, -(len(s+1)), -1)
for ip in range(len(s)-1, -1, -1):
    rs +=s[ip]
print(rs, end = '    ')
```

```
python
nohtyp
```

WAPT print the index positions of even digits present in even index positions.

```
even = '02468'
s = input()
for ip in range(0, len(s),2):
    if s[ip] in even:
        print(s[ip], end = ' ')
```

```
012457836
0 2 8 6
```

Miscellaneous

11 July 2024 04:23 PM

1. WAPT get the output as shown for the given list. (10-July)

```
l = [11,2,44,66,77] # op = ['odd', 'even','even','even', 'odd']
# creating a new list
n = []
for i in l:
    if i%2==0:
        n.append('even')
    else:
        n.append('odd')
print(n)
print()
# creating an existing list
for ip in range(len(l)):
    if l[ip]%2==0:
        l[ip] = 'even'
    else:
        l[ip] = 'odd'
print(l)
```

```
['odd', 'even', 'even', 'even', 'odd']

['odd', 'even', 'even', 'even', 'odd']
```

2. WAPT print the given list as output.

```
ip     ip+1
0  ---> 1 (replace 0th index element with 1 and 1st with 0)
2  ---> 3
4  ---> 5
```

```
l = [11,22,33,44,55,66]
# l = [22,11,44,33,66,55]
for ip in range(0, len(l), 2):
        l[ip], l[ip+1] = l[ip+1], l[ip]

print(l)
```

```
[22, 11, 44, 33, 66, 55]
```

3. Write the output in the below case.

```
# write the output
l = [11,22,33,44,55]
for i in l:
    l.remove(i)
print(i)
print(l)
```

```
55
[22, 44]
```

```
Remove --> will remove the value given as argument, returns None
           when an element is removed, all the succedding values will shift by one index position.
In the for loop, 1st element is removed and control moves to 2nd index position
But element in the 2nd index position has moved to 1st index position, i= 11
so the 2nd index position will now have 3rd value, so i = 33 is removed(after 1st iteration 22 comes to 1st index position)
and so on...
So in such case, every other element will escape from the removal process.
so l = [22,44] are left in the list.
```

4. WAPT print the desired output

```
l = [11,22,33,44,-1,66, 50]
# ['odd', 'even', 'odd', 'even', -1, 'even', 'even']
n = [] # creating a new list
for ip in range(len(l)):
    if l[ip]<0:
        n.append(l[ip])
    elif l[ip] %2 == 0:
        n.append('even')
    else:
        n.append('odd')
print(n, end='\n\n')

# modifying the existing list
for ip in range(len(l)):
    if l[ip]>0:
        if l[ip] %2 == 0:
            l[ip] = 'even'
        else:
            l[ip] = 'odd'
print(l)
```

```
['odd', 'even', 'odd', 'even', -1, 'even', 'even']

['odd', 'even', 'odd', 'even', -1, 'even', 'even']
```

5. Write the output for the following code.

```
l = [11,22,33,44,55]
for l[-1] in l:
    print(l[-1], end = '  ')
print()
print(l)
```

```
11  22  33  44  44
[11, 22, 33, 44, 44]
```

Explanation:

```
In a for loop, generally 1st element is assigned with a variable suppose i
Here 0th element is assigned as l[-1] (--> variable of for loop), so it is l[-1] = 11
Thus the 4th element(55) or l[-1] (--> last element) is replaced by 0th element 11
Next, control moves to 1st index position, l[-1] (--> variable of for loop), = 22
Thus the 4th element(11) or l[-1] (--> last element) is replaced by element in 1st index position 22
Next, control moves to 2nd index position, l[-1] (--> variable of for loop), = 33
Thus the 4th element(22) or l[-1] (--> last element) is replaced by element in 2nd index position 33
Next, control moves to 3rd index position, l[-1] (--> variable of for loop), = 44
Thus the 4th element(44) or l[-1] (--> last element) is replaced by element in 3rd index position 44
Next the control moves to 4th index position, l[-1] (--> variable of for loop), = 44
The 4th element(44) or l[-1] (--> last element) is replaced by element in the 4th index position 44
print l[-1] --> 44
```

```
for l[-1] in l:
    print(l[-1])
    print(l)
    print('*'*20)
print(l)
```

```
11
[11, 22, 33, 44, 11]
********************
22
[11, 22, 33, 44, 22]
********************
33
[11, 22, 33, 44, 33]
********************
44
[11, 22, 33, 44, 44]
********************
44
[11, 22, 33, 44, 44]
********************
[11, 22, 33, 44, 44]
```

```
# Merge the 2 dictionaires below to get the output as shown in new dictionary.
d1 = {'a':20, 'b':30,'c':400,'e':500}
d2 = {'a':40, 'b':20,'c':500,'d':25}
# op = {'a':40, 'b':30,'c':500,'d':25,'e':500}
```

```
op={}

for d in list([d1,d2]):
    for key in d.keys():
        if d != d1 and key not in d1.keys():
            op[key] = d[key]
        else:
            if d1[key]>d[key]:
                op[key] = d1[key]
            else:
                op[key] = d[key]
op
```

```
{'a': 40, 'b': 30, 'c': 500, 'e': 500, 'd': 25}
```

```
op={}
for d in list(d1)+list(d2):
    if d in d1:
        if d in d2:
            if d1[d]>d2[d]:
                op[d]=d1[d]
            else:
                op[d]=d2[d]
        else:
            op[d]=d1[d]
    else:
        op[d]=d2[d]
print(op)
```

```
{'a': 40, 'b': 30, 'c': 500, 'e': 500, 'd': 25}
```

6. Implement the split method. (11 - July)

```
l = []
s = 'hello where are you'
delimiter = ' '
dummy = ''
for i in s:
    if i != delimiter:
        dummy+=i
    else:
        l.append(dummy)
        dummy = ''
l.append(dummy)
print(l)
```

```
['hello', 'where', 'are', 'you']
```

```
iterations:
1. i = 'h'                      13. i = 'a'
dummy = 'h'                         dummy = 'a'
2. i = 'e'                      14. i = 'r'
dummy = 'he'                        dummy = 'ar'
3. i = 'l'                      15. i = 'e'
dummy = 'hel'                       dummy = 'are'
4. i = 'l'                      16. i = ' '
dummy = 'hell'                      l = ['hello', 'where', 'are']
5. i = 'o'                      17. i = 'y'
dummy = 'hello'                     dummy = 'y'
6. i = ' '                      18. i = 'o'
l = ['hello']                       dummy = 'yo'
dummy = ''                      19. i = 'u'
7. i = 'w'                          dummy = 'you'
dummy = 'w'                     20. exit the for loop
8. i = 'h'
dummy = 'wh'
9. i = 'e'
dummy = 'whe'
10. i = 'r'
dummy = 'wher'
11. i = 'e'
dummy = 'where'
12. i= ' '
l = ['hello', 'where']
After all the iterations, the last element in 'dummy' will not beadded to the list,
hence append the last dummy to the list after exiting from the loop.
```

7. Implement the count method

```
itr sub-1  sub-2  sub-3  sub-4   sub-5
0     h      he     hel    hell   hello
1     e      el     ell    ello   ello
2     l      ll     llo    llo    lo
3     l      lo     lo     lo     lo
4     o      o      o      o      o
```

```
s= 'hello'
len(s) = 5 --> total 5 iterations
Supposing that the len(sub) could be anywhere from 0 to len(s), take the range(len(s))
Observe that for sub with length >1, there are unnecessary iterations happening
The number essential iterations for a given string is equal to length of (substring-1)  subtracted from the lengthof
original string.
If the sub == s[ip:ip+len(sub)]: increase the count.
```

```
# implement count method logic
s = 'hello'
c=0
sub = 'l'
for ip in range(len(s) - (len(sub)-1)):#subtract (len(sub)-1) to eliminate the iterations where s[ip : ip+len(sub)]<len(sub)
    if s[ip:ip+len(sub)] == sub:
        c+=1
print(c)
```

```
2
```

Iterations

```
range(5-(1-1))--> range(5)
ip = 0
check s[0:0+1]=s[0:1]='l' == 'h': False
ip = 1
check s[1:1+1]=s[1:2]='l' == 'e': False
ip = 2
check s[2:2+1]=s[2:3]='l' == 'l': True
ip = 3
check s[3:3+1]=s[3:4]='l' == 'l': True
ip = 4
check s[4:4+1]=s[4:5]='l' == 'o': False
exit the loop and print count c = 2
```

WAPT sort a given list without using sort method.

```
# sorting the existing list
def sort_list(l, reverse):
    l=eval(input('Enter a list of values: '))
    for i in range(len(l), 0, -1):
        min_=l[0]
        for n in range(i):
            if reverse ==True:
                if l[:i][n]>min_:
                    min_ = l[n]
            elif l[:i][n]<min_:
                 min_ = l[n]
        l.append(min_)

        for k in range(i):
            if l[k] == l[len(l)-1]:
                l= l[:k]+l[k+1:]
                break
    return l

sort_list(l, False)
```

```
Enter a list of values: [1, 4, 2, 0,8,-5,-1 ]
```

```
[-5, -1, 0, 1, 2, 4, 8]
```

```
# creating a new list
l=[1, 4, 2, 0,8,-5,-1 ]
nl = []
for i in range(len(l), 0, -1):
    max_=l[0]
    for n in range(i):
        if l[:i][n]>max_:
            max_ = l[n]
    l.remove(max_)
    nl.append(max_)
print(nl)
```

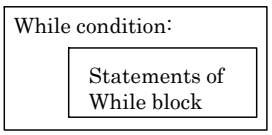
```
[8, 4, 2, 1, 0, -1, -5]
```

While loop

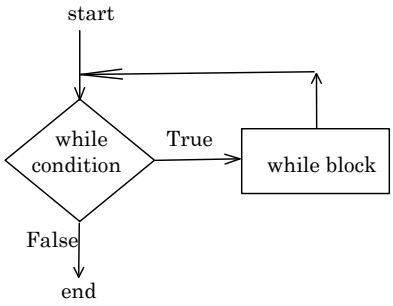
11 July 2024 05:30 PM

1. We use while loop when we are not sure that how many times we have to iterate
2. While loop will be executed based on one condition
3. There is a chance that while loop will be executed for infinite times, in order to avoid that make sure that at one point of time, the condition becomes false.

Syntax:



Flow:



WAPT print the numbers between given number to 10 in reverse order.

```
a=int(input())
print('start')
while a>10:
    print('entered while block', end = '\n')
    print(a, end = '  ')
    a-=1
print('end')
```

```
14
start
entered while block
14  entered while block
13  entered while block
12  entered while block
11  end
```

WAPT print the numbers between given number to 10.

```
a=int(input())
print('start')
while a<10:
    print('entered while block')
    print(a, end = '  ')
    a+=1
print('end')
```

```
8
start
entered while block
8  entered while block
9  end
```

WAPT print the squares of numbers in a given range using both for loop and while loop

```
# using for loop
for i in range(l, u+1):
    print(i**2, end = '  ')

#  using while loop
l = int(input())
u = int(input())
while l<=u:
    print(l**2, end = ' ')
    l+=1
```

```
5
10
25 36 49 64 81 100
```

Note:

- From where while loop begins
  We can create or initialise a variable
- Till where the while loop will run
  By writing in condition
- How the loop will traverse

Either by increment or decrement

HW

1. Write the differences between for loop and while loop

For loop	While loop
For loop is used when the number of iterations is known.	Execution is done in while loop until the statements within the program is proved wrong.
Initialization is done with the help of range and for.	Initialization and increment both are done by the user.
Generally used for a definite number of iterations.	Used for iterations based on a condition which may not be definite.

- Write while loop programs for 1)for loop with range and 2) for loop with range and collection

.....Continued in while loop.ipynb (12-07-24)
.....Continued in Miscellaneous while (with break, continue and pass) - 14-07-24 - incomplete
.....For else, while else (16-07-24) --- check
.....Nested loop(17-07-24/18-07-24)
.....z_quest (all the numbers in 5 ways) (19-07-24) -- check
.....Infinite loop and remaining z_quest(22-07-24) -- check
.....Patterns (25-07-24/26-07-24/29-07-24) -- complete

Deep/shallow/normal copy - explain
While loop - miscellaneous
Miscellaneous while