**Day-01**

**15-03-2025**

**===================**

Python Syntax

Keywords

Identifiers

Variables

Datatypes

Input-Output

Type Conversion

Operators

Conditional Statements

Loop Statements

Transfer Statements

Functions

OOPs

class, Object, Method, Constructor, Destructor and Garbage Collection

Encapsulation, Polymorphism, Inheritance, Abstraction

Exception Handling

File Handling

=================================================================

**Loop Statements:**

**================**

-> Loop: repeat

-> when we need to execute the block of code repeatedly (again and again) based on the condition, we can use "loop statements".

-> There are two types of loops:

1) for loop

2) while loop

1) for loop:

------------

here: "for" is a keyword

which we can use to define the "for" loop

-> to understand the for loop, we need:

1) membership operator: in and not in

2) range() function

-------------------

=> a pre-defined function

which we can use to generate the sequence of numbers based on the values what we have specified.

Ex: 1 to 100, 100 to 200

Syntax-1: range(one parameter)

-------------------------------------

range(value)

Ex: range(10)

here:

here: 10 can understand as "stop value".

range() can generate values up to 10 (by excluding 10)

start value by default as '0'

range(10) ==> 0 to 9

a = range(10)

print(a)

print(range(20))

print(list(a))

print(list(range(20)))

Syntax-2: range(With two parameters)

--------------------------------------------

range(start, stop)

Ex: range(1,10)

here:

start = 1

stop = 10

-> range() can start to generate the values from '1' to (10-1) '9'

a = list(range(1,10))

print(a)

Syntax-3: range(with three parameters)

----------------------------------------------

range(start, stop, step)

Here:

step ==> the difference from current value to the next value

Ex: range(10,100,10)

==> start = 10

stop = 100

step = 10

==> 10, 20, 30, 40, 50, 60, 70, 80, 90

a = list(range(10,100,10))

print(a)

Syntax of the for loop:

--------------------------

for variable/iterable in range(parameters):

statement-1

statement-2

statement-3

**# Write a python program to print "hi" for 10 times**

for i in range(10):

print("Hi")

**Q: Is it possible to print multiple values within the same line.**

Ans: yes

using "end" keyword in print()

Syntax:

print(value, end = "")

**# Write a python program to print n-natural numbers.**

# natural numbers ==> 1, 2, 3, .... n

n = int(input("Enter length of natural number series:"))

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

print(i, end = "\t")

**# Write a python program to print the sum of n-natural numbers.**

# solution-1: using formulae

"""

n\*(n+1)/2

"""

n = int(input("Enter the length of the natural number series:"))

result = (n \* (n+1))//2

print("The sum of",n,"natural numbers is = ",result)

# solution-2: using for loop

s = 0

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

s = s + i # 1 3 6

print("The sum of",n,"natural numbers is = ",s)

Assignment:

--------------

1) Write a python program to print all even natural numbers and odd natural numbers both separately.

2) Write a python program to print the sum of all even natural numbers and all odd numbers using for loop.

**Day-02**

**18-03-2025**

**=================**

**Nested for loop:**

**===========**

-> Writing of a for loop inside the another for loop is called as "Nested for loop".

Syntax:

for variable1 in range(start1, stop1, step1):

for variable2 in range(start2, stop2, step2):

statement-1

statement-2

statement-3

statement-4

1) Initially, in outer for loop the variable1 can generate the first value like (s), with that value the control enter into the loop body of the outer for loop.

2) In the loop body of outer for loop, the inner for loop can start variable2 and assume it can generate a value 's1' and with that value the control can enter loop body of the inner for loop and make execute the statements like 1 and 2 after that the control can back to range() and it can generate the next value as per the given range. With that value again the inner for loop body can start execute.

Repeat this until the last value from the range of inner for loop. Once all the values of inner for loop range is completed in generation then the control can go to statements of outer for loop body like 3 and 4 for the execution.

3) Then the control can back to range() in outer for loop. Then again it can repeat the above steps of operation until the last value.

**# nested for loop demo**

for i in range(1,6):

print(i)

for j in range(1,6):

print(j,end = "\t")

print()

**# Write a python program to print all multiplication tables from 10 to 21.**

"""

1 table

2 table

3 table

....

10 table

"""

for number in range(10,21):

print("Table for:",number)

for i in range(1,11):

print(number,"X",i,"=",number\*i)

print()

**# Write a python program to print all perfect numbers from 1 to 500.**

"""

Perfect number: Sum of all the factors of the given number excluding itself is equals to

given number is called as "Perfect number"

Ex: 48 ==> 1, 2, 3, 4, 6, 8, 12, 16, 24, 48

1 + 2 + 3 + 4 + 6 + 8 + 12 + 16 + 24 ==> 78

78 == 48 ==> not a perfect number

6 ==> 1, 2, 3, 6

1 + 2 + 3 ==> 6

6 == 6 ==> perfect numbers

"""

for i in range(1,501):

s = 0

for j in range(1,i):

if i % j == 0:

s = s + j

if s == i:

print(i,end = "\t")

**Day-03**

**19-03-2025**

**======================**

**Write a python program to print all the prime numbers from the given range.**

Ans:

----

range ==> 1 to 20

prime number ==> have the factors are only two (1 and itself)

factor ==> m ==> 1 and m are only the factors ==> prime numbers

m % 2 == 0 ==> 2 is the factor for m

Note:

-----

1 is not a prime number

2 is the least and only even prime number

2

2 to 19

17

17 % 2 != 0

17 % 3 != 0

17 % 4 != 0

17 % 5 != 0

17 % 6 != 0

17 % 7 != 0

17 % 8 != 0

17 % 9 != 0

......

17 % 16 != 0

17 ==> prime number

7 ==> 2 to 6

2,3, 5, 7, 11, 13, 17, 19

**Transfer Statements:**

**==============**

-> also called as "Jump Statements".

-> three different types of transfer statements:

1) break

2) continue

3) return

1) break:

----------

-> break is the keyword

-> break always allowed to define within the loops only.

-> when we want to terminate the program/loop immediately based on the certain condition, we can use "break".

# 1 generate the range of numbers to check which are prime numbers

n1 = int(input("Enter the first value of the range:"))# 10

n2 = int(input("Enter the second value of the range:")) # 50

for ravi in range(n1,n2+1):

flag = False

for prime in range(2,ravi):

if ravi % prime == 0:

flag = True

break

if flag == False:

print(ravi,end = "\t")

Problem-2:

------------

for i in range(1,11):# 1 to 10 (7 to 10)

if i == 7:

break

print(i,end = "\t")

2) continue:

-------------

-> is a keyword

-> continue also can use within the loops only.

-> when we want to skip the current iteration and continue with remaining, then we can use "continue" statement.

# 1 generate the range of numbers to check which are prime numbers

n1 = int(input("Enter the first value of the range:"))# 10

n2 = int(input("Enter the second value of the range:")) # 50

for ravi in range(n1,n2+1):

if ravi == 1:

continue

flag = False

for prime in range(2,ravi): # range(2,2)

if ravi % prime == 0:

flag = True

break

if flag == False:

print(ravi,end = "\t")

Problem-3:

-------------

for i in range(1,11):

if i == 7:

continue

print(i,end = "\t")