Chapter-8: Python Control Statements or Control Structures

Control Statements or Control Structures

Content

1. Conditional Statements
    a. If
        i. Simple if
        ii. If..else
        iii. If..elif..else
        iv. Nested if
    b. Match
2. Looping Control Statements
    a. while
    b. for
    c. Nested looping statements
3. Branching statements
    a. Break
    b. Continue
    c. Pass

Objective of learning control statements or control structures to handle/control the flow of execution of a program.

Conditional Control Statements or Conditional Statements

Conditional control statements or conditional statements are used to execute blocks of statements based on condition.

Python supports 2 types of conditional control statements

1. If
2. match (python 3.10)

if statement

"if" is a keyword, which represents a conditional control statement.
If statement is used to execute a block of statements based on condition.

Python support different types of if statements

1. simple if
2. if..else
3. if..elif..else
4. nested if

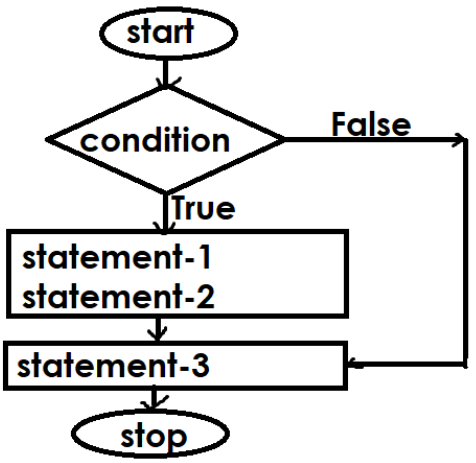
simple if

if without else is called simple if.
This syntax is having only if block without else block.

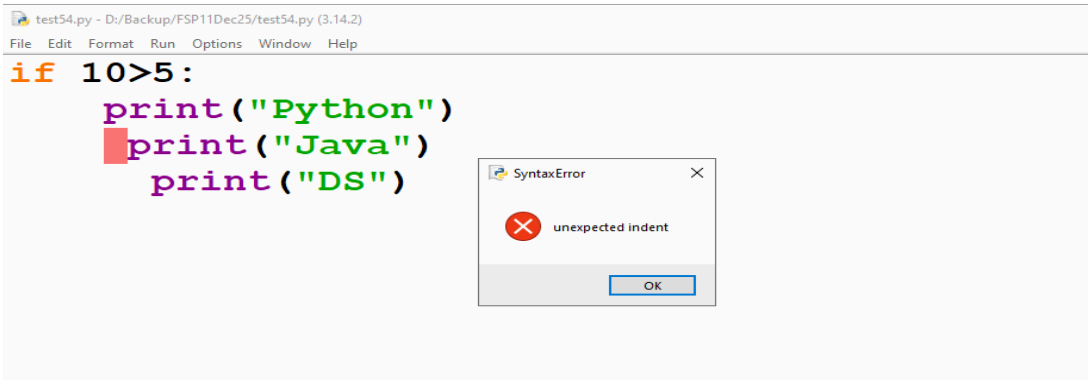
Syntax of simple if

```
if  condition:
  statement-1
  statement-2
statement-3
```

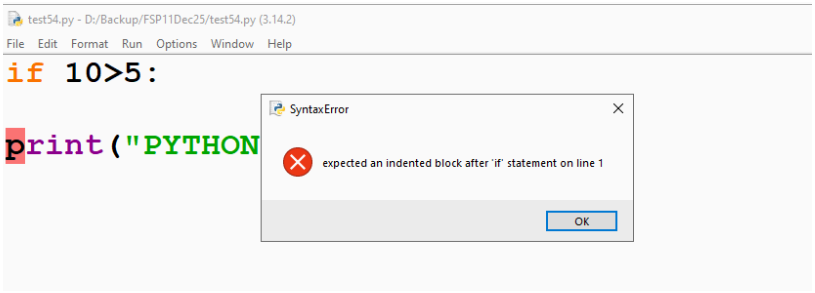
if condition is True, python executes statement-1, statement-2 and statement-3
if condition is False, python continue statement-3



if True:
  print("PYTHON")
 	
if False:
  print("JAVA")
 	
if True:
 print("DS")
	Output
	PYTHON
	DS



The above code generate syntax error (unexpected indent)
All the statements inside block must be at same indentation level (same spaces)



The above code generates syntax error (expected an indented block after if statement).
In python a block must have at least one statement (OR) python does not allows empty blocks.

“pass” keyword OR “pass” statement

This statement is written inside blocks.
In python empty blocks are represented using “pass” keyword”
“pass” indicates do-nothing or null operation.

Example:

```
if 10>5:
    pass

print("PYTHON")
print("JAVA")
print("DS")
```

Output

PYTHON
JAVA
DS

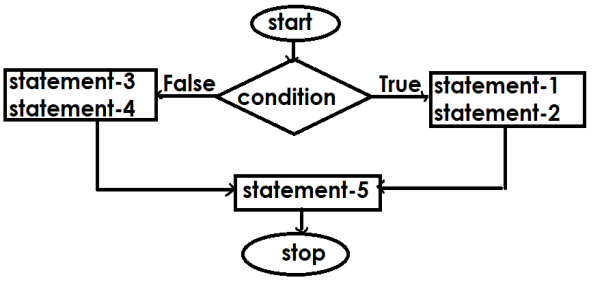
if..else block

This syntax is having 2 blocks

1. if block
2. else block

Syntax:

```
if condition:
   Statement-1
   Statement-2
else:
   statement-3
   statement-4
statement-5
```



if condition is True, python runtime executes statement-1,statement-2 and continue statement-5 if condition is False, python runtime executes statement-3,statement4 and continue with statement5	

Example
if True:
  print("if block")
else:
  print("else block")
 	
if False:
  print("if block")
else:
  print("else block")
	Output
	if block
	else block

```
# Write a program to check wather a person is
# eligible to vote or not

name=input("Name :")
age=int(input("Age :"))

if age>=18:
    print(f'{name} is eligible to vote')
else:
    print(f'{name} is not eligible to vote')
```

Output

```
Name :naresh
Age :40
naresh is eligible to vote
```

Name :suresh
Age :12
suresh is not eligible to vote

```
# Write a program to check whether number
# entered by user is even or odd

num=int(input("Enter any number "))
if num%2==0:
   print(f'{num} is Even')
else:
   print(f'{num} is Odd')
```

Output

```
Enter any number 8
8 is Even

Enter any number 15
15 is Odd
```

```
# Write a program to check whether a number
# is divisible by 7 or not

num=int(input("Enter any number "))
if num%7==0:
   print(f'{num} is divisible')
else:
   print(f'{num} is not divisible')
```

```
# Write a program to display "Hello"
# if a number entered by user is a multiple of five
# otherwise print "Bye"



num=int(input("Enter any number "))
if num%5==0:
```

```
    print("Hello")
else:
    print("Bye")
```

Output

```
Enter any number 30
Hello

Enter any number 32
Bye
```

https://www.codechef.com/problems/MANGOLASSI

```
temp=int(input())
if temp>35:
    print("YES")
else:
    print("NO")
```

https://www.codechef.com/problems/CHEFPAROLE

```
days=int(input())
if days>=7:
    print("YES")
else:
    print("NO")
```

https://www.codechef.com/problems/ADVITIYA1

```
day=int(input())
if day==16 or day==17 or day==18:
    print("ADVITIYA")
else:
```

```
    print("WAITING FOR ADVITIYA")
```

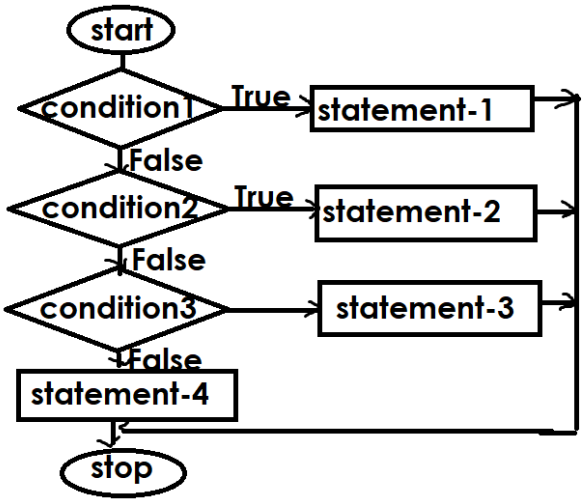
https://www.codechef.com/problems/ACPLZ

```
T=int(input())
if T>30:
    print("YES")
else:
    print("NO")
```

If..elif..else (if..else ladder)

This syntax allows checking more than one condition.

```
if condition1:
   statement-1
 elif condition2:
    statement-2
elif condition3:
    statement-3
....
else:
  statement-x
```



if condition1 is True, python runtime executes statement-1
if condition1 is False and condition2 is True, python runtime execute statement-2
if condition1,condition2 are False and condition3 is True, python runtime executes statement-3

if all conditions are False, python runtime execute optional else block which contains statement-x	

https://www.codechef.com/problems/HEALSE

```
hours=int(input())
if hours<8:
    print("LESS")
elif hours==8:
    print("PERFECT")
else:
    print("MORE")
```

Q1. Write a program to accept percentage from the user and display the grade according to the following criteria:

Marks	Grade
> 90	A
> 80 and <= 90	B
>= 60 and <= 80	C
below 60	D

Show Answer

```
p=int(input("Marks Percentage"))
if p>90:
    print("A Grade")
elif p>80 and p<=90:
    print("B Grade")
elif p>=60 and p<=80:
    print("C Grade")
```

```
else:
    print("D Grade")
```

Q2. Write a program to accept the cost price of a bike and display the road tax to be paid according to the following criteria :

Cost price (in Rs)	Tax
> 100000	15 %
> 50000 and <= 100000	10%
<= 50000	5%

Show Answer

```
price=int(input("Bike Cost :"))
if price>100000:
    tax=price*15/100
elif price>50000 and price<=100000:
    tax=price*10/100
else:
    tax=price*5/100

print(f'Bike Cost {price}')
print(f'Road Tax {tax}')
```

 # Write a program to find max of 3 numbers

```
n1=int(input("enter first number :"))
n2=int(input("enter second number:"))
n3=int(input("enter third number :"))
if n1>n2 and n1>n3:
    print(f'{n1} is max')
elif n2>n1 and n2>n3:
    print(f'{n2} is max')
elif n3>n1 and n3>n2:
    print(f'{n3} is max')
```

Q8. Write a program to calculate the electricity bill (accept number of unit from user) according to the following criteria :

Unit	Price
First 100 units	no charge
Next 100 units	Rs 5 per unit
After 200 units	Rs 10 per unit

(For example if input unit is 350 than total bill amount is Rs2000)

```
units=int(input("Units :"))
if units<=100:
    bill=0
elif units>100 and units<=200:
    bill=(units-100)*5
else:
    bill=0+500+(units-200)*10


print(f'Units {units}')
print(f'Bill {bill}')
```

Q3. A company decided to give bonus to employee according to following criteria:

Time period of Service	Bonus
More than 10 years	10%
>=6 and <=10	8%
Less than 6 years	5%

Ask user for their salary and years of service and print the net bonus amount.

```
year=int(input("Years of service :"))
salary=int(input("Salary :"))
if year>10:
    bonus=salary*10/100
elif year>=6 and year<=10:
    bonus=salary*8/100
else:
    bonus=salary*5/100
```

```
print(f'Years of Service {year}')
print(f'Salary {salary}')
print(f'Bonus {bonus}')
```

https://www.hackerrank.com/challenges/py-if-else/problem?isFullScreen=true

```
n=int(input())
if n%2!=0:
    print("Weird")
elif n>=2 and n<=5:
    print("Not Weird")
elif n>=6 and n<=20:
    print("Weird")
else:
    print("Not Weird")
```

Nested if

If inside if is called nested if

Nested if is used to split and operation or (or) operation into multiple if statements.

Syntax

```
if condition1:
   if condition2:
      statement-1
   else:
      statement-2
else:
  statement-3
```

Example:

```
#Login Application

user=input("UserName :")
pwd=input("Password :")
if user=="nit":
   if pwd=="n123":
    print("Welcome")
   else:
    print("invalid password")
else:
    print("invalid username")
```

Output

```
UserName :nit
Password :n123
Welcome

UserName :nit
Password :n321
invalid password

UserName :xyz
Password :abc
invalid username
```

Example:

```
# Banking Application
# Logic of withdraw/deposit
accno=1
cname="naresh"
balance=50000
ttype=input("Transaction Type :")
tamt=int(input("Transaction Amount :"))
if ttype=="deposit":
```

```
    balance=balance+tamt
elif ttype=="withdraw":
    if balance>tamt:
        balance=balance-tamt
    else:
        print("Insuff balance")
else:
    print("invalid transaction type")

print(f"'AccountNo {accno}
CustomerName {cname}
Balance {balance}"')
```

Output

```
Transaction Type :withdraw
Transaction Amount :15000
AccountNo 1
CustomerName naresh
Balance 35000

Transaction Type :withdraw
Transaction Amount :70000
Insuff balance
AccountNo 1
CustomerName naresh
Balance 50000
```

Example:

```
# Write a program to find the max of 3 numbers
# using nested if

n1=int(input("Enter First Number :"))
n2=int(input("Enter Second Number :"))
n3=int(input("Enter Third Number :"))
```

```
if n1>n2:
    if n1>n3:
        print(f'{n1} is max')
    else:
        print(f'{n3} is max')
elif n2>n1:
    if n2>n3:
        print(f'{n2} is max')
    else:
        print(f'{n3} is max')
elif n3>n1:
    if n3>n2:
        print(f'{n3} is max')
    else:
        print(f'{n2} is max')
```

Output

```
Enter First Number :30
Enter Second Number :10
Enter Third Number :20
30 is max

Enter First Number :10
Enter Second Number :20
Enter Third Number :10
20 is max
```

match statement

“match” statement is introduced in the Python 3.10 version.
“match” is a softkeyword.
“match” statement is similar to switch statement in other programming languages (C,C++,Java,.Net)

“match” statement is called a pattern matching statement.

Syntax:

```
match subject:
        case pattern1:
                statement-1
                statement-2
        case pattern2:
                statement-3
                statement-4
        case pattern3:
                statement-5
                statement-6
        …
        case _:
                statement-x
```

Here, the subject is an expression whose value is compared against the patterns in each case block. The first pattern that matches has its associated code block executed. The underscore (_) acts as a wildcard and will match if no other case does, serving as a default.

The match statement supports various types of patterns.

1. **Matching Literals:** You can match against constant values like numbers or strings.

```
num=int(input("Enter any number "))
match num:
   case 1:
      print("One")
   case 2:
      print("Two")
   case 3:
```

```
        print("Three")
    case 4:
        print("Four")
    case 5:
        print("Five")
    case _:
        print("invalid number")
```

Output

```
Enter any number 3
Three

Enter any number 9
invalid number
```

Example:

```
print("Course Details")
print("1. Java")
print("2. Python")
print("3. .Net")
print("4. Oracle")
print("5. Exit")
opt=int(input("Enter Your Option "))
match opt:
    case 1:
        print(f"'Java Fee : 2000
Duration:3months"')
    case 2:
        print(f"'Python Fee : 8000
Duration:3Months"')
    case 3:
        print(f"'.Net Fee :4000
Duration:3Months"')
    case 4:
```

```
        print(f'''Oracle Fee:2000
Duration:3Months''')
    case 5:
        print("Bye...")
    case _:
        print("Invalid Option Try Again")
```

Output

```
Course Details
1. Java
2. Python
3. .Net
4. Oracle
5. Exit
Enter Your Option 9
Invalid Option Try Again

Course Details
1. Java
2. Python
3. .Net
4. Oracle
5. Exit
Enter Your Option 4
Oracle Fee:2000
Duration:3Months
```

2. **Combining Patterns:** The | operator allows you to combine multiple literals in a single case. This is useful for handling several possible inputs with the same code block.

```
n=4
match n:
    case 1|3|5:
```

```
        print("Hello")
    case 2 | 4 | 6:
        print("Hi")
    case _:
        print("Bye")
```

Output

Hi

```
# Write a program to find given alphabet is vowel or not

ch=input("Alphabet ")
match ch:
    case 'a' | 'e' | 'o' | 'u' | 'i':
        print("Vowel")
    case 'A' | 'E' | 'O' | 'U' | 'I':
        print("Vowel")
    case _:
        print("Not Vowel")
```

Output

Alphabet a
Vowel

Alphabet I
Vowel

3. **Adding Conditions (Guards):** An if clause can be included in a case to specify an additional condition that must be true for the pattern to match. This "guard" is checked only after the initial pattern is successful.

Example:

```
num=15
```

```
match num:
    case 15 if num%2==0:
        print("Hello")
    case 15 if num%2!=0:
        print("Hi")
```

Output

Hi

Example:

```
# Calculator
n1=int(input("Enter First Number :"))
n2=int(input("Enter Second Number :"))
opr=input("Enter Operator(+,-,/,*) :")
match opr:
    case '+':
        print(f'sum of {n1} and {n2} is {n1+n2}')
    case '-':
        print(f'diff of {n1} and {n2} is {n1-n2}')
    case '*':
        print(f'product of {n1} and {n2} is {n1*n2}')
    case '/' if n2!=0:
        print(f'division of {n1} and {n2} is {n1/n2}')
    case '/' if n2==0:
        print("cannot divide number with zero")
    case _:
        print("invalid operator")
```

Output

```
Enter First Number :10
Enter Second Number :2
Enter Operator(+,-,/,*) :/
division of 10 and 2 is 5.0
```

```
Enter First Number :10
Enter Second Number :0
Enter Operator(+,-,/,*) :/
cannot divide number with zero
```

4. **Matching and Deconstructing Data Structures:** match can analyze the structure of data like lists, tuples, or dictionaries and bind their components to variables

Example:

```
stud=[1,"naresh","python"]
match stud:
    case [1,"naresh","java"]:
        print("naresh with java")
    case [1,"naresh","python"]:
        print("naresh with python")
    case _:
        print("not matched student found")
```

Output

naresh with python

Looping Control Statements

Looping control statements are used to repeat a block of statements a number of times or until a given condition.
Python support 2 types of looping control statements

1. while loop
2. for loop

while loop

"while" is a keyword, which represents a while loop in python.
"while" loop is used to repeat a block of statements until the given condition is True.

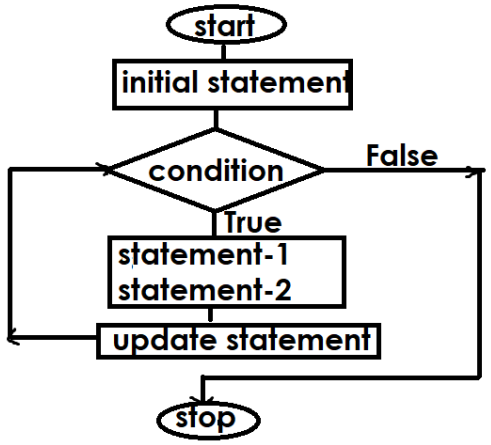
Syntax-1: only while	**Syntax-2: while..else**

while condition:
  statement-1
  statement-2
 	
Statement-1, Statement-2 is repeated until given condition is True, if condition is False, it stop repeating statement-1,statement-2
	while condition:
	  statement-1
	  statement-2
	else:
	  statement-3
	  statement-4

While loop required 3 statements

1. initialization statement
2. Condition/Boolean expression
3. Update Statement

Initialization statement: This statement defines the initial value of condition.
Condition: Condition is a boolean expression which defines how many times the loop has to be repeated.
Update statement: statement which updates condition



i=1
while i<=5:
  print("Hello")
  i=i+1
	Hello
	Hello
	Hello
	Hello
	Hello

# Write a program to input 5 numbers and print sum 	
 	
i=1
s=0
while i<=5:
  num=int(input("Enter any number:"))
  s=s+num
  i=i+1
 	
print(f'Sum is {s}')
	Enter any number:10
	Enter any number:20
	Enter any number:30
	Enter any number:40
	Enter any number:50
	Sum is 150

# Write a program to generate numbers from 1 to 10 	
# using while loop 	
 	
num=1
while num<=10:
  print(num,end=' ')
  num=num+1
	1 2 3 4 5 6 7 8 9 10

# write a program to generate numbers from 10 to 1 	
# using while loop 	
 	
num=10
 	
while num>=1:
  print(num,end=' ')
  num=num-1
	10 9 8 7 6 5 4 3 2 1

# Write a program to generate numbers from -1 to -10 	
 	
num=-1
while num>=-10:
  print(num,end=' ')
  num=num-1
	-1 -2 -3 -4 -5 -6 -7 -8 -9 -10

# Write a program to generate math table of a given number 	
 	
num=int(input("Enter any number "))
i=1
while i<=10:
  p=num*i
  print(f'{num}x{i}={p}')
  i=i+1
	Enter any number 10
	10x1=10
	10x2=20
	10x3=30
	10x4=40
	10x5=50
	10x6=60
	10x7=70
	10x8=80
	10x9=90
	10x10=100

# Write a program to find length of number (OR) 	
# Count digits 	
 	
num=int(input("Enter any number "))
c=0
while num>0:
  c=c+1
  num=num//10
 	
print(f'Count of Digits {c}')
	Enter any number 489
	Count of Digits 3
	
	Enter any number 37934
	Count of Digits 5

# Write a program to input a number and find # sum of digits # 125 ==> 1+2+5 ==> 8	Enter any number 125 Sum of Digits 8

num=int(input("Enter any number "))
s=0
while num>0:
 	
  r=num%10
  s=s+r
  num=num//10
 	
print(f'Sum of Digits {s}')	

# Write a program to count even and odd digits in a given 	
# number 	
# 3785 	
# even digits=1 	
# odd digits=3 	
 	
num=int(input("Enter any number "))
ec=0
oc=0
 	
while num>0:
  r=num%10
  if r%2==0:
  ec+=1
  else:
  oc+=1
  num=num//10
 	
print(f'Even Digits Count {ec}')
print(f'Odd Digits Count {oc}')
	Enter any number 1235
	Even Digits Count 1
	Odd Digits Count 3

# Write a program to find input 3 digit number is 	
# armstrong or not 	
 	
num=int(input("enter any number "))
num1=num
 	
s=0
while num>0:
  r=num%10
  s=s+(r**3)
  num=num//10
 	
if s==num1:
  print(f'{num1} is armstrong number')
else:
  print(f'{num1} is not armstrong number')
	enter any number 153
	153 is armstrong number
	
	enter any number 126
	126 is not armstrong number

# Write a program to reserve the input number 	
 	
# 123 ==> 321 	
 	
num=int(input("enter any number "))
num1=num
rev=0
 	
while num>0:
  r=num%10
  rev=(rev*10)+r
  num=num//10
	Output
	enter any number 123
	321
	not palindrome

print(rev)
if rev==num1:
  print("palindrome")
else:
  print("not palindrome")	

# Write a program to convert decimal number to binary number 	
 	
num=int(input("enter any number "))
binary=""
while num>0:
  r=num%2
  binary=binary+str(r)
  num=num//2
 	
print(binary[::-1])
	enter any number 12
	1100

# Write a program to convert decimal number to 	
# hexadecimal number 	
 	
num=int(input("enter any number "))
hexa=""
 	
while num>0:
  r=num%16
  match r:
  case 10:
  hexa=hexa+"a"
  case 11:
	enter any number 255
	0xff
	
	enter any number 26
	0x1a

```
            hexa=hexa+"b"
        case 12:
            hexa+="c"
        case 13:
            hexa+="d"
        case 14:
            hexa+="e"
        case 15:
            hexa+="f"
        case _:
            hexa=hexa+str(r)
    num=num//16

print("0x"+hexa[::-1])
```

 # Write a program to find factorial of given number

```
num=int(input("Enter any number "))
i=1
fact=1
while i<=num:
    fact=fact*i
    i=i+1

print(f'Factorial of {num} is {fact}')
```

Output

Enter any number 4
Factorial of 4 is 24

Enter any number 5
Factorial of 5 is 120

Example:

```
# Write a program to generate fibonacci series
'''
The Fibonacci series is a sequence of numbers
where each number is the sum of the two preceding ones,
typically starting with 0 and 1,
creating the pattern: 0, 1, 1, 2, 3, 5, 8, 13, and so on'''


a=0
b=1
c=0
print(a,b,end=' ')
while c<13:
    c=a+b
    print(c,end=' ')
    a=b
    b=c
```

Output

```
0 1 1 2 3 5 8 13
```

for loop

"for" is a keyword, which represents a for loop in python.
"for" loop in python is used to iterate values from iterables/collections and executing blocks of statements.

Without else	**With else**

Syntax-1:
 	
for variable in iterable/collection:
  statement-1
  statement-2
	Syntax-2:
	
	for variable in iterable/collection:
	  statement-1
	  statement-2
	else:

variable: hold value read from iterable or collection for loop each time read value from collection and execute block of statements. Statement,statement-2 are executed until read all the values from iterable or collection	statement-3 statement-4

Example
for a in "PYTHON":
print(a)
 	
for b in [10,20,30,40,50]:
print(b)
 	
for x in range(10):
print("PYTHON")
	Output
	P
	Y
	T
	H
	O
	N
	10
	20
	30
	40
	50
	PYTHON
	PYTHON
	PYTHON
	PYTHON
	PYTHON
	PYTHON
	PYTHON
	PYTHON
	PYTHON
	PYTHON

range datatype

range is an immutable sequence data type.
range data type is used for generating sequence of integer values.
The range type represents an immutable sequence of numbers and is commonly used for looping a specific number of times in for loops.

Syntax-1: range(stop)
Syntax-2: range(start,stop,step)

In application development range data type is used for,

1. Repeating for loop number of times
2. For generating values for other collections

range data type is used for generating a sequence of integer values in increment or decrement order.

range data type required 3 inputs

1. Start
2. Stop
3. Step

Start: an integer value which represents starting value of range
Stop: an integer value which represents ending/stop value of range, which is not included (excluded)
Step: an integer value which represents difference between value (increment or decrement value). This value can be +ve or –ve but should not be zero.

Syntax-1: range(stop)
This syntax create range with default start=0,step=1

Example:

```
a=range(5) #start=0,stop=5,step=1
for num in a: # 0 1 2 3 4
    print(num)

for num in range(10): #start=0,stop=10,step=1
    print(num,"Hello")    #0 1 2 3 4 5 6 7 8 9
```

Output

```
0
1
2
3
4
0 Hello
1 Hello
2 Hello
3 Hello
4 Hello
5 Hello
6 Hello
7 Hello
8 Hello
9 Hello
```

This syntax allows to generate sequence +ve integer values.

```
for num in range(-5): # start=0,stop=-5,step=1
    print(num)
```

Output:
No output

range data type always generate values based on step value

1. If step is +ve, start<stop
2. If step is –ve,start>stop

Syntax2: range(start,stop,step=1)
This syntax allows to generate sequences of integers in +ve range and –ve range.

Example:

```
# generate numbers from 1 to 10

for num in range(1,11,1):
    print(num,end=' ')

print()
# generate even numbers 2 4 6 8 10 12 14 16 18 20
for num in range(2,21,2):
    print(num,end=' ')

print()
# generate odd numbers 1 3 5 7 9 11 13 15 17 19
for num in range(1,20,2):
    print(num,end=' ')

print()
# generate number 1 4 7 10 13 16 19
for num in range(1,22,3):
    print(num,end=' ')

print()
# generate numbers 10 9 8 7 6 5 4 3 2 1
for num in range(10,0,-1):
    print(num,end=' ')

print()
```

```
# generate numbers 10 8 6 4 2
for num in range(10,0,-2):
    print(num,end=' ')

print()
# generates numbers -1 -2 -3 -4 -5 -6 -7 -8 -9 -10
for num in range(-1,-11,-1):
    print(num,end=' ')

print()
# generate numbers -10 -9 -8 -7 -6 -5 -4 -3 -2 -1
for num in range(-10,0,1):
    print(num,end=' ')
print()
# generate numbers 5 4 3 2 1 0 -1 -2 -3 -4 -5

for num in range(5,-6,-1):
    print(num,end=' ')
print()

# generate numbers from -5 -4 -3 -2 -1 0 1 2 3 4 5
for num in range(-5,6,1):
    print(num,end=' ')

print()
# generate Alphabets from A B C D E F ... Z

for num in range(65,91):
    print(chr(num),end=' ')

print()
# generate Alphabets from a b c d e f ... z
for num in range(97,123):
    print(chr(num),end=' ')
```

```
print()
# generate the series 1 4 9 16 25 36 49 81 100
for num in range(1,11):
    print(num**2,end=' ')
```

Output

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
1 3 5 7 9 11 13 15 17 19
1 4 7 10 13 16 19
10 9 8 7 6 5 4 3 2 1
10 8 6 4 2
-1 -2 -3 -4 -5 -6 -7 -8 -9 -10
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1
5 4 3 2 1 0 -1 -2 -3 -4 -5
-5 -4 -3 -2 -1 0 1 2 3 4 5
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
1 4 9 16 25 36 49 64 81 100
```

Example:

```
# Write a program to generate math table of given number

num=int(input("Enter any number "))

for i in range(1,11): # 1 2 3 4 5 6 7 8 9 10
    p=num*i
    print(f'{num}x{i}={p}')
```

Output

```
Enter any number 5
5x1=5
5x2=10
```

5x3=15
5x4=20
5x5=25
5x6=30
5x7=35
5x8=40
5x9=45
5x10=50

 # Write a program to find input number is prime or not

```
num=int(input("enter any number "))
c=0
for i in range(1,num+1):
    if num%i==0:
        c=c+1

if c==2:
    print(f'{num} is prime')
else:
    print(f'{num} is not prime')
```

Output

```
enter any number 7
7 is prime

enter any number 9
9 is not prime
```

 # Write a program to input n integer values
 # count even and odd number

```
n=int(input("Enter how many values?"))
```

```
even_count=0
odd_count=0
for i in range(n):
    value=int(input("Enter Integer Value :"))
    if value%2==0:
        even_count=even_count+1
    else:
        odd_count=odd_count+1

print(f'Even Numbers Count {even_count}')
print(f'Odd Numbers Count {odd_count}')
```

Output

```
Enter how many values?5
Enter Integer Value :5
Enter Integer Value :2
Enter Integer Value :9
Enter Integer Value :1
Enter Integer Value :4
Even Numbers Count 2
Odd Numbers Count 3
```

 # Write a program to input n values
 # and count of integer values,float values

```
n=int(input("Enter how many values?"))
c1=0
c2=0
c3=0
for i in range(n):
    value=eval(input("Enter any value "))
    if isinstance(value,(int,)):
        c1=c1+1
    elif isinstance(value,(float,)):
        c2=c2+1
```

```
    else:
        c3=c3+1

print(f'Count of integer values {c1}')
print(f'Count of float values {c2}')
print(f'Count of other type of values {c3}')

# Write a program to input n integer values
# find minimum value, maximum value
n=int(input("Enter how many values ?"))
min_value=0
max_value=0
x=True
for i in range(n):
    value=int(input("Enter value :"))
    if x:
        min_value=value
        max_value=value
        x=False
    elif value>max_value:
        max_value=value
    elif value<min_value:
        min_value=value

print(f'Maximum value {max_value}')
print(f'Minimum value {min_value}')
```

Output

```
Enter how many values ?5
Enter value :10
Enter value :5
Enter value :8
Enter value :9
Enter value :11
```

Maximum value 11
Minimum value 5
Write a program to find factorial of a given number

```
num=int(input("enter any number "))
fact=1
for i in range(1,num+1):
    fact=fact*i

print(f'Factorial of {num} is {fact}')
```

Output

enter any number 4
Factorial of 4 is 24

enter any number 5
Factorial of 5 is 120

How to input multiple values in single line?

```
s=input()    "1 2 3 4"

L=s.split() --> ["1","2","3","4"]

A,B,C,D=L

s1=input()   "1,2,3,4"
L=s1.split(",")      ["1","2","3","4"]
    a,b,c,d=map(int,L)
```

```
s=input()
L=s.split()
a,b,c,d=L
print(L)
print(a,b,c,d,type(a),type(b),type(c),type(d))
a,b,c,d=map(int,L)
print(a,b,c,d,type(a),type(b),type(c),type(d))
```

Output

```
1 2 3 4
['1', '2', '3', '4']
1 2 3 4 <class 'str'> <class 'str'> <class 'str'> <class 'str'>
1 2 3 4 <class 'int'> <class 'int'> <class 'int'> <class 'int'>
```

https://www.codechef.com/problems/GOLDCOINS

```
A,B,X,Y=map(int,input().split())
if X>Y:
    print(A)
else:
    print(B)
```

https://www.codechef.com/problems/HEATWAVE

```
t1,t2=map(int,input().split())
if t2>t1:
    print("YES")
else:
    print("NO")
```

https://www.codechef.com/problems/CANDIVIDE

```
T=int(input())

for i in range(T):
    candies=int(input())
    if candies%3==0:
        print("YES")
    else:
        print("NO")
```

https://www.codechef.com/problems/BTRYHLTH

```
T=int(input())

for i in range(T):
    X=int(input())
    if X>=80:
        print("YES")
    else:
        print("NO")
```

Branching Statements or Passes Control Statements

Python supports 2 branching statements

1. break
2. continue

Branching statements are used to control the execution of looping control statements (while, for)

break statement

“break” is a keyword which represents branching statements in python.

This statement is used to terminate execution of while loop or for loop in between or unconditionally.

This statement is used only inside for loop or while loop.

Example:

```
for i in range(10):
    print("Inside for loop")
    break

i=1
while i<=10:
    print("Inside while loop")
    i=i+1
    break
```

Output

```
Inside for loop
Inside while loop
```

Example:

```
# Login with OTP number
import random
otp=random.randint(1000,9999)
print(f"Please Type OTPNO {otp} to Login")
res=False
for i in range(3):
    iotp=int(input("Type OTPNO :"))
    if otp==iotp:
        print("Login Completed")
        res=True
        break
    else:
        print("Invalid OTP try again")

if res==False:
    print("Account is Locked completed 3 attempts")
```

Output

```
Please Type OTPNO 3340 to Login
Type OTPNO :3341
Invalid OTP try again
Type OTPNO :3342
Invalid OTP try again
Type OTPNO :3343
Invalid OTP try again
Account is Locked completed 3 attempts
```

Example:

```
# Write a program to find input number is prime or not
```

```
num=int(input("Enter any number "))
i=1
c=0
res=True
while i<=num:
    r=num%i
    if r==0:
        c=c+1
    if c>2:
        res=False
        break
    i=i+1

if res:
    print(f'{num} is prime')
else:
    print(f'{num} is not prime')
```

Output

```
Enter any number 5
5 is prime

Enter any number 6
6 is not prime
```

Example:

```
# Write a program read/input numbers until number is -ve
# number

while True:
    num=int(input("Enter any number "))
    if num<0:
        break
```

Output

Enter any number 87
Enter any number 45
Enter any number 34
Enter any number -1

Example:

```
#MENU
# Calculator
while True:
    print("1.Add")
    print("2.Sub")
    print("3.Multiply")
    print("4.Division")
    print("5.Exit")
    opt=int(input("Enter Your Option:"))
    if opt==1 or opt==2 or opt==3 or opt==4:
        n1=int(input("Enter First Number :"))
        n2=int(input("Enter Second Number :"))

    if opt==1:
        print(f'Sum of {n1} and {n2} is {n1+n2}')
    elif opt==2:
        print(f'Diff of {n1} and {n2} is {n1-n2}')
    elif opt==3:
        print(f'Product of {n1} and {n2} is {n1*n2}')
    elif opt==4:
        print(f'Division of {n1} and {n2} is {n1/n2}')
    elif opt==5:
        break
    else:
        print("invalid option try again...")
```

Output

```
1.Add
2.Sub
3.Multiply
4.Division
5.Exit
Enter Your Option:3
Enter First Number :5
Enter Second Number :2
Product of 5 and 2 is 10
1.Add
2.Sub
3.Multiply
4.Division
5.Exit
Enter Your Option:6
invalid option try again...
```

Example:

```
# Write a program to print first n integer numbers
# division with 3

n=int(input("How many integer values?"))
num=3
while True:
    if num%3==0:
        print(num,end=' ')
        n=n-1

    if n==0:
        break
    num=num+1
```

Output

How many integer values?10
3 6 9 12 15 18 21 24 27 30

continue

"continue" is a keyword which represents a branching statement in python.

This statement is used inside looping control statements like while loop or for loop.

"continue" statement or keyword, continue execution of while loop or for loop without executing rest of the statements.

Syntax-1:
 	
while condition:
  statement-1
  continue
  statement-2
	Syntax-2:
	
	for variable-name in iterable:
	  statement-1
	  continue
	  statement-2

for i in range(5):
  print("Hello")
  continue
  print("Bye")
 	
i=1
while i<=5:
  i=i+1
  print("Hello")
  continue
  print("Bye")
	Output
	Hello
	Hello
	Hello
	Hello
	Hello
	Hello
	Hello
	Hello
	Hello
	Hello

# Find output for n in range(1,21):	**Output** **1** **3**

if n%2==0:
  continue
print(n)
 	
n=1
while n<=10:
  if n%2==0:
  n=n+1
  continue
  print(n)
  n=n+1
	5
	7
	9
	11
	13
	15
	17
	19
	1
	3
	5
	7
	9

while..else (OR) for..else

Syntax of while..else	**Syntax of for..else**

while <condition>:
  Statement-1
  Statement-2
else:
  Statement-3
  Statement-4
	for variable-name in iterable:
	  statement-1
	  statement-2
	else:
	  statement-3
	  statement-4

While block executed until given condition is True, when the condition is False, it executes the else block and stops. Else block is not execute if while loop is terminated using "break" statement.	for block is executed until read all the values from iterable/collection. After reading all the value, it executes else block else block is not executed if for loop is terminated using break statement.

for i in range(5):
  print("For Block")
else:
  print("Else Block")
 	
i=1
while i<=5:
  print("While Block")
  i=i+1
else:
  print("Else Block")
	Output
	For Block
	For Block
	For Block
	For Block
	For Block
	Else Block
	While Block
	While Block
	While Block
	While Block
	While Block
	Else Block

for i in range(5):
  print("For Block")
  break
else:
  print("Else Block")
 	
i=1
while i<=5:
  print("While Block")
  i=i+1
  break
else:
  print("Else Block")
	Output
	For Block
	While Block

# Login with OTPNO import random OTP=random.randint(1000,9999) print(f'Type OTPNO {OTP} to Login') for i in range(3):	**Output** **Type OTPNO 8826 to Login** **Input OTPNO:8824** **Invalid OTP Try Again...** **Input OTPNO:8821** **Invalid OTP Try Again...** **Input OTPNO:8825**

iotp=int(input("Input OTPNO:"))
if iotp==OTP:
print("Login Completed")
break
else:
print("Invalid OTP Try Again...")
else:
print("Account is Blocked 3 Attempts complted")
	Invalid OTP Try Again...
	Account is Blocked 3 Attempts complted

# Prime Number 	
num=int(input("Enter any number :"))
i=1
c=0
while i<=num:
if num%i==0:
c=c+1
if c>2:
print(f'{num} is not prime')
break
i=i+1
else:
print(f'{num} is prime')
	Output
	Enter any number :5
	5 is prime
	
	Enter any number :8
	8 is not prime

Nested Looping Statements

Defining a looping control statement within a looping control statement is called nested looping control statements.

Nested looping control statements are 2 types

1. Nested for loop
2. Nested while loop

Nested for loop

for loop inside for loop is called nested for loop

Syntax:

for variable-name in iterable/collection: Outer Loop
    for variable-name in iterable/collection: Inner Loop
        statement-1
        statement-2
    statement-3

Example:	**Output**

for i in range(5): # 0 1 2 3 4
  for j in range(3): # 0 1 2
  print("Hello")
	Hello
	Hello
	Hello
	Hello
	Hello
	Hello
	Hello
	Hello
	Hello
	Hello
	… 15 times

# Write a program to generate math tables from 1 to 10 	
 	
for num in range(1,11): # 1 2 3 4 5 6 7 8 9 10
  for i in range(1,11): # 1 2 3 4 5 6 7 8 9 10
  p=num*i
  print(f'{num}x{i}={p}')
	Output
	1x1=1
	1x2=2
	1x3=3
	1x4=4
	1x5=5
	1x6=6
	1x7=7
	1x8=8
	1x9=9
	1x10=10

	2x1=2 2x2=4 2x3=6 2x4=8 … 10 tables

# Write a program to generate prime numbers from 2 to 30 	
 	
for num in range(2,31): # 2 3 4 5 6 7 8 9 10 ... 30
  c=0
  for i in range(1,num+1):
  if num%i==0:
  c=c+1
  if c==2:
  print(f'PrimeNo:{num}')
	Output
	PrimeNo:2
	PrimeNo:3
	PrimeNo:5
	PrimeNo:7
	PrimeNo:11
	PrimeNo:13
	PrimeNo:17
	PrimeNo:19
	PrimeNo:23
	PrimeNo:29

# Write a program to generate factorails of 	
# all numbers from 0 to 5 	
 	
for num in range(0,6): # 0 1 2 3 4 5
  fact=1
  for i in range(1,num+1):
  fact=fact*i
  print(f'Factorial of {num} is {fact}')
	Output
	Factorial of 0 is 1
	Factorial of 1 is 1
	Factorial of 2 is 2
	Factorial of 3 is 6
	Factorial of 4 is 24
	Factorial of 5 is 120

for i in range(3):
  for j in range(3):
  print(i,j)
	Output
	0 0
	0 1
	0 2

	1 0 1 1 1 2 2 0 2 1 2 2

Nested While loop

While loop inside while loop is called nested while loop

Syntax:

while condition: □ Outer while loop
    while condition: □ Inner while loop
        statement-1
        statement-2
    statement-3

i=1
while i<=2: # Outer Loop
  j=1
  while j<=2: # Inner Loop
  print("Hello")
  j=j+1
  i=i+1
	Output
	Hello
	Hello
	Hello
	Hello

# Write a program to generate 	
math tables from 1 to 10
# using nested while loop 	
 	
num=1
while num<=10:
  i=1
  while i<=10:
  p=num*i
  print(f'{num}x{i}={p}')
	Output
	1x1=1
	1x2=2
	1x3=3
	1x4=4
	1x5=5
	1x6=6
	1x7=7
	1x8=8
	1x9=9

i=i+1 num=num+1	1x10=10 2x1=2 2x2=4 2x3=6 2x4=8 … 10 tables

# Write a program to generate armstrong numbers from 	
# 100 to 999 	
 	
num=100
while num<=999:
num1=num
s=0
while num1>0:
d=num1%10
s=s+(d**3)
num1=num1//10
 	
if s==num:
print(num)
num=num+1
	Output
	153
	370
	371
	407

# Write a program to generate palindrom numbers from 	
# 100 to 200 	
 	
num=100
while num<=200:
rev=0
num1=num
while num1>0:
d=num1%10
rev=(rev*10)+d
	Output
	101
	111
	121
	131
	141
	151
	161
	171
	181
	191

num1=num1//10 if num==rev: print(num) num=num+1	