Chapter 10
Conditional & Looping Constructs

Computer Science
Class XI (As per CBSE Board)

Visit: python.mykvs.in for regular updates
Control statements are used to control the flow of execution depending upon the specified condition/logic.

There are three types of control statements.

1. Decision Making Statements
2. Iteration Statements (Loops)
3. Jump Statements (break, continue, pass)
Decision Making Statement

Decision making statement used to control the flow of execution of program depending upon condition.

There are three types of decision making statement.

1. if statements
2. if-else statements
3. Nested if-else statement
An if statement is a programming conditional statement that, if proved true, performs a function or displays information.
1. if statements

Syntax:

```python
if(condition):
    statement
    [statements]
```

E.g.

```python
noofbooks = 2
if (noofbooks == 2):
    print('You have ')
    print('two books')
print('outside of if statement')
```

Output

```
You have two books
```

Note: To indicate a block of code in Python, you must indent each line of the block by the same amount. In above e.g. both print statements are part of if condition because of both are at same level indented but not the third print statement.

Visit: python.mykvs.in for regular updates
2. if-else Statements

#find absolute value

```python
a=int(input("enter a number"))
if(a<0):
    a=a*-1
print(a)
```

#it will always return value in positive
1. if statements

Using logical operator in if statement

```python
x=1
y=2
if(x==1 and y==2):
    print('condition matching the criteria')
```

Output :-
condition matching the criteria

```python
a=100
if not(a == 20):
    print('a is not equal to 20')
```

Output :-
a is not equal to 20
2. if-else Statements

If-else statement executes some code if the test expression is true (nonzero) and some other code if the test expression is false.
2. if-else Statements

Syntax:

```python
if(condition):
    statements
else:
    statements
```

e.g.

```python
a=10
if(a < 100):
    print('less than 100')
else:
    print('more than equal 100')
```

**OUTPUT**

less than 100

*Write a program in python to check that entered number is even or odd*
3. Nested if-else statement

The nested if...else statement allows you to check for multiple test expressions and execute different codes for more than two conditions.
3. Nested if-else statement

Syntax:
If (condition):
    statements
elif (condition):
    statements
else:
    statements

E.G.
num = float(input("Enter a number: "))
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")

OUTPUT
Enter a number: 5
Positive number

* Write python program to find out largest of 3 numbers.

Visit: python.mykvs.in for regular updates
3. Nested if-else Statements

sort 3 numbers

```python
first = int(input("Enter the first number: "))
second = int(input("Enter the second number: "))
third = int(input("Enter the third number: "))
small = 0
middle = 0
large = 0
if first < third and first < second:
    small = first
    if second < third and second < first:
        small = second
    else:
        small = third
elif first < second and first < third:
    middle = first
    if second > first and second < third:
        middle = second
    else:
        middle = third
elif first > second and first > third:
    large = first
    if second > first and second > third:
        large = second
    else:
        large = third
print("The numbers in accending order are: ", small, middle, large)
```

Visit: python.mykvs.in for regular updates
3. Nested if-else Statements

# Check leap year / divisibility

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

if (year % 4) == 0:
    if (year % 100) == 0:
        if (year % 400) == 0:
            print("{0} is a leap year".format(year))
        else:
            print("{0} is not a leap year".format(year))
    else:
        print("{0} is a leap year".format(year))
else:
    print("{0} is not a leap year".format(year))
```

Visit: python.mykvs.in for regular updates
Iteration statements (loop) are used to execute a block of statements as long as the condition is true. Loops statements are used when we need to run same code again and again.

Python Iteration (Loops) statements are of three type:

1. While Loop
2. For Loop
3. Nested For Loops
1. While Loop

It is used to execute a block of statement as long as a given condition is true. And when the condition become false, the control will come out of the loop. The condition is checked every time at the beginning of the loop.

Syntax

```python
while (condition):
    statement
    [statements]
```

E.g.

```python
x = 1
while (x <= 4):
    print(x)
    x = x + 1
```

Output

```
1
2
3
4
```
While Loop continue

While Loop With Else

e.g.

x = 1
while (x < 3):
    print('inside while loop value of x is ', x)
    x = x + 1
else:
    print('inside else value of x is ', x)

Output
inside while loop value of x is 1
inside while loop value of x is 2
inside else value of x is 3

*Write a program in python to find out the factorial of a given number
While Loop continue

Infinite While Loop

e.g.
x = 5
while (x == 5):
    print('inside loop')

Output
Inside loop
Inside loop
...
...
2. For Loop

It is used to iterate over items of any sequence, such as a list or a string.

Syntax

```python
for val in sequence:
    statements
```

e.g.

```python
for i in range(3,5):
    print(i)
```

Output

3
3
4
2. For Loop continue

Example programs

```python
for i in range(5, 3, -1):
    print(i)
```

Output
5
4

range() Function Parameters

- **start**: Starting number of the sequence.
- **stop**: Generate numbers up to, but not including this number.
- **step** (Optional): Determines the increment between each numbers in the sequence.
Example programs with range() and len() function

fruits = ['banana', 'apple', 'mango']
for index in range(len(fruits)):
    print('Current fruit :', fruits[index])

range() with len() Function Parameters
2. For Loop continue

For Loop With Else

e.g.
for i in range(1, 4):
    print(i)
else:  # Executed because no break in for
    print("No Break")

Output
1
2
3
No Break
2. For Loop continue
   Nested For Loop

   e.g.
   for i in range(1,3):
       for j in range(1,11):
           k=i*j
           print (k, end=' ')
       print()
   print()

Output
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
2. For Loop continue

Factorial of a number

factorial = int(input('enter a number'))

# check if the number is negative, positive or zero
if num < 0:
    print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    for i in range(1,num + 1):
        factorial = factorial*i
    print("The factorial of",num,"is",factorial)
2. For Loop continue

Compound Interest calculation

```python
n=int(input("Enter the principle amount:"))
rate=int(input("Enter the rate:"))
years=int(input("Enter the number of years:"))

for i in range(years):
    n=n+((n*rate)/100)
print(n)
```

Visit : python.mykvs.in for regular updates
Jump statements are used to transfer the program's control from one location to another. Means these are used to alter the flow of a loop like - to skip a part of a loop or terminate a loop.

There are three types of jump statements used in python.

1. break
2. continue
3. pass
1. break

It is used to terminate the loop.

e.g.
for val in "string":
    if val == "i":
        break
    print(val)

print("The end")

Output
s
t
r
The end

Visit : python.mykvs.in for regular updates
It is used to skip all the remaining statements in the loop and move controls back to the top of the loop.

e.g.
for val in "init":
    if val == "i":
        continue
    print(val)
print("The end")

Output
n
t
The end
3. pass Statement
This statement does nothing. It can be used when a statement is required syntactically but the program requires no action.

Use in loop
while True:
    pass  # Busy-wait for keyboard interrupt (Ctrl+C)

In function
It makes a controller to pass by without executing any code.
e.g.
def myfun():
    pass  #if we don’t use pass here then error message will be shown
print('my program')

OUTPUT
My program
3. pass Statement continue
e.g.
for i in 'initial':
    if(i == 'i'):
        pass
    else:
        print(i)

OUTPUT
n
ta
L

NOTE: continue forces the loop to start at the next iteration while pass means "there is no code to execute here" and will continue through the remainder or the loop body.