DATE : 11 june 2024

DAY : Tuesday

TOPICS : Conditional Statements and loops

Conditional Statements

Conditional Statements are statements in Python that provide a choice for the control flow based on a condition. It means that the control flow of the Python program will be decided based on the outcome of the condition.

⌄ Types of Conditional Statements

1. IF STATEMENT

2. IF...ELSE STATEMENT

3. IF..ELIF...ELSE STATEMENT

4. NESTED IF STATEMENT

5. NESTED IF ELSE STATEMENT

1. IF STATEMENT*

If the simple code of block is to be performed if the condition holds then the if statement is used. Here the condition mentioned holds then the code of the block runs otherwise not.

Syntax of If Statement:

```
#if condition:
    # Statements to execute if
    # condition is true
```

```
n1 = int(input("Enter a number:"))
if n1 % 2==0:
  print("You have entered an even number")
print("done!")
```

```
⇥   Enter a number:5
    done!
```

```
age=int(input("enter your age:"))
if age>18:
  print("you can apply for license")
print("speed thrills but kills")
```

```
⇥   enter your age:20
    you can apply for license
    speed thrills but kills
```

2. IF...ELSE STATEMENT

In a conditional if Statement the additional block of code is merged as an else statement which is performed when if condition is false.

Syntax of If-Else Statement:

```
#if (condition):
    # Executes this block if
    # condition is true
#else:
    # Executes this block if
    # condition is false
```

```
a = int(input("Enter a positive number:"))
if a>0:
  print("Well done!")
```

```
else:
  print("Try again.")
```

```
⊋▾   Enter a positive number:6
     Well done!
```

```
age=int(input("enter your age : "))
if age>18:
  print("you can apply for license")
else:
  print("not valid")
```

```
⊋▾   enter your age : 20
     you can apply for license
```

3. IF..ELIF...ELSE STATEMENT

The if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed. If none of the conditions is true, then the final "else" statement will be executed.

Syntax of IF..ELIF...ELSE STATEMENT

```
#if condition1:
    # execute body if condition 1 is true.
 #elif condition2:
    #......
 #elif condition3:
    #.....
 #else:
    # executed if none of above conditions are satisfied.
```

```
n = int(input("Your marks:"))
if n>=80 and n<=90:
 print("Grade A")
elif n>70 and n<80:
 print("Grade B")
elif n>60 and n<70:
 print("Grade Ck")
elif n>50 and n<60:
 print("Grade D")
else:
 print("Sorry, you are not qualified.")
```

```
⊋▾   Your marks:89
     Grade A
```

```
age =44

if age < 13:
    category = 'Child'
elif age < 20:
    category = 'Teenager'
elif age < 65:
    category = 'Adult'
else:
    category = 'Senior'

print(f"The person is classified as: {category}")
```

```
⊋▾   The person is classified as: Adult
```

4. NESTED IF STATEMENT

if statement can also be checked inside other if statement. This conditional statement is called a nested if statement. This means that inner if condition will be checked only if outer if condition is true and by this, we can see multiple conditions to be satisfied.

Syntax of NESTED IF STATEMENT

```
#if (condition1):
    # executes when condition is True
 # if (condition2):
        # executes when condition is True
```

```
gpa = int(input("Enter your gpa:"))
if gpa>7.5:
  print("Congratulations!, you are eligible.")
  b = int(input("Enter no. of pending backlogs:"))
  if b==0:
    print("Well done, Go ahead!")
    mat = int(input("Enter your matriculation percentage:"))
    if mat>75:
      print("Just one step more.")
      sec = int(input("Enter you 12th percentage:"))
      if sec>60:
        print("Congratulations! you are in.")
        print("Better luck next time.")
```

```
Enter your gpa:8
Congratulations!, you are eligible.
Enter no. of pending backlogs:0
Well done, Go ahead!
Enter your matriculation percentage:90
Just one step more.
Enter you 12th percentage:98
Congratulations! you are in.
Better luck next time.
```

5. NESTED IF ELSE STATEMENT

Nested if..else means an if-else statement inside another if statement. Or in simple words first, there is an outer if statement, and inside it another if − else statement is present and such type of statement is known as nested if statement. We can use one if or else if statement inside another if or else if statements.

Syntax of NESTED IF ELSE STATEMENT

```
#if condition 1:
    #if condition 2:
        #statement(s)
    #else:
        #statement(s)
#else:
    #statement(s)
```

```
age=int(input("enter your age"))
own_car='false'
if(age>=18):
  if(own_car=="true"):
    print("drive your car")
  else:
      print("work hard and purchase a new car")
else:
      print("not required")
```

˅ Practice Questions

You have a variable age. Write a conditional statement to check if age is greater than or equal to 18. If true, print "You are an adult", otherwise print "You are a minor"

```
age = 18
if age>=18:
  print("you are adult")
else:
  print("you are minor")
```

```
you are adult
```

Given a variable temperature, write a conditional statement to check if temperature is below 0. If true, print "It's freezing", otherwise print "It's not freezing".

```
temperature = int(input("enter the temperature :"))
if temperature<0:
  print("it is  freezing")
```

```
else:
  print("it is not freezing")
```

```
enter the temperature :-3
it is  freezing
```

You have a variable score. Write a conditional statement to check if score is greater than 90. If true, print "Grade A", if score is between 80 and 90, print "Grade B", otherwise print "Grade C".

```
score = int(input("enter the score : "))
if score>90:
  print("grade A")
elif score<= 90 and score>= 80:
  print("grade B ")
else:
  print("grade C")
```

```
enter the score : 67
grade C
```

Given two variables a and b, write a conditional statement to check if a is equal to b. If true, print "a and b are equal", otherwise print "a and b are not equal".

```
a = 4
b = 7
if a==b:
  print("a and b are equal")
else:
  print("a and b are not equal")
```

```
a and b are not equal
```

You have a variable number. Write a conditional statement to check if number is even or odd. If even, print "Even number", otherwise print "Odd number".

```
number = 6
if 6%2 == 0:
  print("even")
else:
  print("odd")
```

```
even
```

Given a variable day which can be any day of the week, write a conditional statement to print "Weekend" if the day is "Saturday" or "Sunday", otherwise print "Weekday".

```
day = input("Enter the day : ")
if day == "saturday" or "sunday":
  print("weekend")
else:
  print("weekday")
```

```
Enter the day : saturday
weekend
```

You have a variable marks. Write a conditional statement to check if marks are greater than 75. If true, print "Distinction", if marks are between 50 and 75, print "Pass", otherwise print "Fail".

```
marks = int(input("Enter the marks : "))
if marks>75:
  print("distiction")
elif marks<=75 and marks>=50:
  print("pass")
else:
  print("fail")
```

```
Enter the marks : 70
pass
```

You have a variable year. Write a conditional statement to check if year is a leap year. If true, print "Leap year", otherwise print "Not a leap year".

```
year = int(input("enter the year : "))
if year%4 == 0:
  print("leap year")
else:
  print("non-leap year")
```

```
enter the year : 1999
non-leap year
```

Given a variable char, write a conditional statement to check if char is a vowel (a, e, i, o, u). If true, print "Vowel", otherwise print "Consonant".

```
str = input("enter the character ")
vowels = "aeiou"
if str in vowels:
  print("vowel")
else:
  print("consonant")
```

```
enter the character a
vowel
```

You have two variables x and y. Write a conditional statement to check if both x and y are positive. If true, print "Both are positive", otherwise print "At least one is not positive".

```
x = int(input("Enter the first number : "))
y = int(input("Enter the second number : "))
if x>=0 and y>=0:
  print("both are positive")
else:
  print("at least one is negative ")
```

```
Enter the first number : 3
Enter the second number : 7
both are positive
```

Given a variable time representing the hour of the day in 24-hour format, write a conditional statement to print "Good morning" if time is between 6 and 12, "Good afternoon" if time is between 12 and 18, and "Good evening" if time is between 18 and 24.

```
time = int(input("enter the time "))
if time>=6 and time<12:
  print("good morning")
elif time>=12 and time<18:
  print("good afternoon")
elif time>=18 and time<24:
  print("good evening")
```

```
enter the time 18
good evening
```

Given a variable username, write a conditional statement to check if username is not empty. If true, print "Username is valid", otherwise print "Username cannot be empty".

```
username = int(input("Enter the username "))
if username == 0:
  print("username can not be empty")
else:
  print("username is valid")
```

```
Enter the username 453
username is valid
```

Given a variable month, write a conditional statement to check if month is "December", "January", or "February". If true, print "Winter", if month is "June", "July", or "August", print "Summer", otherwise print "Other season".

```
month = input("Enter the month ")
if month == "december" or month == "january" or  month == "february":
  print("winter")
elif month =="june"or month == "july" or month == "august":
  print("summer")
```

```
Enter the month january
winter
```

Given a variable password, write a conditional statement to check if the length of password is greater than or equal to 8. If true, print "Strong password", otherwise print "Weak password"

```
password = input("enter the password ")
if len(password)>=8:
  print("strong password")
else:
  print("weak password")
```

```
enter the password 12fgf234
strong password
```

You have a variable weight and a variable height. Write a conditional statement to calculate BMI and print "Underweight" if BMI is less than 18.5, "Normal weight" if BMI is between 18.5 and 24.9, and "Overweight" if BMI is 25 or above.

```
weight = int(input("Enter the weight "))
height = int(input("Enter the height "))
bmi = weight/height
if bmi<=18.5:
  print("Underweight")
elif bmi>18.5 and bmi<=24.5:
  print("normal weight")
elif bmi>= 25:
  print("overweight")
```

```
Enter the weight 45
Enter the height 135
Underweight
```

LOOPS

Loops are important in Python or in any other programming language as they help you to execute a block of code repeatedly. You will often come face to face with situations where you would need to use a piece of code over and over but you don't want to write the same line of code multiple times.

In Python we have mainly two different types of loops :

for loop : In the context of most data science work, Python for loops are used to loop through an iterable object (like a list, tuple, set, etc.) and perform the same action for each entry. For example, a for loop would allow us to iterate through a list, performing the same action on each item in the list.

while loop : The while loop is somewhat similar to an if statement, it executes the code inside, if the condition is True. However, as opposed to the if statement, the while loop continues to execute the code repeatedly as long as the condition is True.

⌄ for loops

A for loop acts as an iterator in Python; it goes through items that are in a sequence or any other iterable item. Iterable is an object, which one can iterate over. Objects that we've learned about that we can iterate over include strings, lists, tuples, and even built-in iterables for dictionaries, such as keys or values.

Here's the general format for a for loop in Python:

```python
#for item in object:
    # statements to do stuff


for i in range(1,11):
  print(i)


for i in range(1,11):
  print(i**2)


list1=[1,2,3,4]
# list1[0]
for i in range(0,len(list1)):
  print(list1[i])


str1="Prabhgun"
for i in range(0,len(str1)):
  print(str1[i])


num=int(input('Enter a number: '))
if num==0:
  print(1)
elif num==1:
  print(1)
else:
  fact=1
  for i in range(1,num+1):
    fact=fact*i
print(fact)


l1=[1,2,3,4]
add=0
for i in l1:
  add=add+i
  # print(add)
print(add)
```

The variable name used for the item is completely up to the coder, so use your best judgment for choosing a name that makes sense and you will be able to understand when revisiting your code. This item name can then be referenced inside your loop, for example if you wanted to use if statements to perform checks.

Let's go ahead and work through several example of for loops using a variety of data object types

Example 1

Let us print each element of our list of strings using a for loop statement

```python
# Consider a list of strings
got_houses = ['Stark', 'Arryn', 'Baratheon', 'Tully', 'Greyjoy', 'Lannister', 'Tyrell', 'Martell', 'Targaryen']


 #A simple for loop to print the houses of GOT universe
for house in got_houses[::-1]:
  print(f"House {house}")
```

Another interesting way to loop through the elements of a list is to use the enumerate() function. Using enumerate requires us two iterators index and element

```python
got_houses
```

```python
x = 1,2
x
```

```python
for j in range(0,3):
    print(j)
```

```python
# Nested loop
for i in range(0,4):
  for j in range(0,3):
    print(j)
```

```python
string="AmanDeepSingh"
# string
for i in string:
  print(i)
```

```python
string="AmanDeepSingh"
# string
for i in range(0,len(string)):
  print(string[i])
```

Example 2

Suppose you are given a list of numbers. You need to find the corresponding squares of these numbers and zip them together in a dictionary.

```python
# The list of numbers
list_of_numbers = [1, 2, 4, 6, 11, 14, 17, 20]
```

```python
even=[]
odd=[]
for i in list_of_numbers:
  if i%2==0:
    even.append(i)
  else:
    odd.append(i)

print(even)
print(odd)
```

```python
for i in list_of_numbers:
  if i%2==0:
    print("even")
  else:
    print("odd")
```

```python
# Let us first print the squares
for number in list_of_numbers:
  squared_number = number**2
  print(f"The square of {number} is {squared_number}")
```

So this was a pretty straight forward way to print out these squares.

Example 3

Imagine a scenario where we not only needed to print these numbers for each iteration but also we need to store these elements somewhere else

```python
list_of_numbers
```

```python
# Let us first initialize a list where we will be appending the squares in each iteration

squared_numbers = []

for number in list_of_numbers:
  square = number**2
  # Use the append method to add the numbers one by one to our list
  squared_numbers.append(square)
  # print(squared_numbers)

print(f"The list of squared numbers is {squared_numbers}")

print(list_of_numbers)
print(squared_numbers)
```

```
# Let us zip the original numbers and squares together and get the dictionary
zipped_dict = dict(zip(list_of_numbers,squared_numbers))

# Let us print the dictionary where the key is the number and the value is the square of that number
print(zipped_dict)
```

∨ While Loops

The while statement in Python is one of the most general ways to perform iteration. A while statement will repeatedly execute a single statement or group of statements as long as the condition is true. The reason it is called a 'loop' is because the code statements are looped through over and over again until the condition is no longer met.

The general format of a while loop is:

```
#while test:
    #code statements
#else:
    #final code statements


x = 0

while x < 10:
  print('x is currently: ',x)
  # print('x is still less than 10, adding 1 to x')
  x=x+1
```

Notice how many times the print statements occurred and how the while loop kept going until the False condition was met, which occurred once x==10. It's important to note that once this occurred the code stopped. Let's see how we could add an else statement:

```
x = 0

while x < 5:
  print('x is currently: ',x)
  print(' x is still less than 10, adding 1 to x')
  x+=1

print("All done")
print("I am done with the iterations")
```

∨ break, continue, pass

We can use break, continue, and pass statements in our loops to add additional functionality for various cases. The three statements are defined by:

break: Breaks out of the current closest enclosing loop.

continue: Goes to the top of the closest enclosing loop.

pass: Does nothing at all.

Thinking about break and continue statements, the general format of the while loop looks like this:

```
#while test:
    #code statement
    #if test:
        #break
    #if test:
        #continue
#else:
```

∨ break and continue statements can appear anywhere inside the loop's body, but we will usually put them further nested in conjunction with an if statement to perform an action based on some condition.

```
x = 0

while x < 10:
  print('x is currently: ',x)
  x=x+1
  if x==3:
    print('x==3')
    break
  else:
    print('continuing...')
    continue
```

Note how we have a printed statement when x==3, and a continue being printed out as we continue through the outer while loop. Let's put in a break once x ==3 and see if the result makes sense:

```
x = 0

while x < 10:
  print('x is currently: ',x)
  print(' x is still less than 10, adding 1 to x')
  x+=1
  break

else:
  print('continuing...')
```

```
list2=[12,4,5,67,8,9,10]
for i in list2:
  if i%2==0:
    pass
  else:
    print(i)
```

```
5
67
9
```

Practice Questions

You are managing a list of employee ages. Print each age from the list using a for loop.

```
employee_ages = [34,45,29,31,44]
for i in range(0,len(employee_ages)):
  print(f"Age : {employee_ages[i]} ")
```

```
Age : 34
Age : 45
Age : 29
Age : 31
Age : 44
```

You have a string representing a secret message. Use a while loop to print each character until you reach a period (.).

```
str = "Hi,my name is Jasmine."
i = 0
while i<len(str):
  print (str[i],end = '')
  i+=1
```

```
Hi,my name is Jasmine.
```

You are given a list of daily temperatures. Use a for loop to calculate and print the average temperature for the week.

```
temperatures = [25, 70, 35, 64, 45, 75, 22]
length = len(temperatures)
sum= 0.0
for i in range(0,length):
  sum = sum + temperatures[i]
average = sum/length
print(f"average temperature : {average}")
```

    →   average temperature : 48.0

You are processing a list of book titles. Use a for loop to print each title in uppercase.

```
book_title ='pride and prejudice,power of subconcious mind,the lord of the rings'

for i in range(0,len(book_title)):
    print(book_title[i].upper(),end='')
```

    →   PRIDE AND PREJUDICE,POWER OF SUBCONCIOUS MIND,THE LORD OF THE RINGS

You are counting items in a warehouse. Use a for loop to skip printing the count for the item labeled "defective", but print counts for all other items.

```
warehouse = {'football':'undefective','basketball':'defective','cricket bat':'defective','badminton racket':'undefective'}
#items = list(warehouse)
#count = 0
for i in range(0,len(warehouse)):
  if "defective" in warehouse:
    break

  else:
    print(len(warehouse))
   # count+= 1
    #print(num_of_item)
```

    →   4
         4
         4
         4

You are programming a countdown timer. Write a while loop to print the countdown from 10 to 0.

```
#countdown = [10,9,8,7,6,5,4,3,2,1,0]
timer = 10
while timer>=0:
  print(timer)
  timer-=1
```

    →   10
         9
         8
         7
         6
         5
         4
         3
         2
         1
         0

You are analyzing sales data. Use a for loop to find and print the highest sale amount from a list of sales figures.

```
sales_figure = [1999, 3900,3000,2000,1000,4777]
highest = 0
for i in range(1,len(sales_figure)):
  if sales_figure[i]>highest:
```

```
      highest = sales_figure[i]
print(f"The highest sales amount : $ {highest}")
```

    The highest sales amount : $ 4777

You are maintaining a guest list. Use a for loop to print each guest's name, but stop if you encounter the name "VIP".

```
guest_name = ["Ema","William","Johns","Carol", "James"  , "Ellie"]
for i in range(0,6):
  if 'VIP' in guest_name:
    print("VIP encountered in the list of guest names")
    break
  else:
    print(guest_name[i])
```

    Ema
    William
    Johns
    Carol
    James
    Ellie

Start coding or generate with AI.