Day_2_Control_Flow

February 4, 2023

1 4) Control Flow

Here, we are going to review and learn the most fundamentals of Python control flow: Decision making statement and loops.

1.1 4.1 Decision Making Statement

- It is kind of making decision during occurred situation of program execution and action can be taken according to specified conditions.
- Structure of decision making evaluate several expressions that provide True or False as a result.
- It is up to you to decide which type of action want to take and execute the statements based upon True and False.

These are several topics of decision making which going to be discussed below: 1. **if** 2. **if else** 3. **if elif else** 4. **Nested if** 5. **Single Statement Suites**

1.1.1 4.1.1 If statement

- In python `if` statement is same as other languages.
- It is having a logical expression where data is getting compared and based upon comparison result a decision is made.

Examples

[ ]:
```
'''Its a simple if statement,
if the value is defined then it
display the print statements'''

val=84
if val:
    print("Value")
    print(val)
```

```
Value
84
```

Exercise

[ ]:
```

```

1.1.2 4.1.2 If-else statement

- With `if` statement an `else` statement is combined.
- `if-else` statement have a block of code, where `if` is executed when it is `1` or `True` and `else` when `if` is `0` or `False` >**Note:** The `else` statement is consider to be an optional statement, so there could be only one `else` statement in block of code.

Examples

[ ]:
```
'''Its a good example to get
the Net payable amount by setting up
discount at specific price range by using if-else statement'''

Total_Price=int(input("Price: "))
if Total_Price<100:
    discount=Total_Price*0.04
    print ("Discount Rupess: "+str(discount))
else:
    discount=Total_Price*0.1
    print ("Discount Rupess: "+str(discount))
print ("Net payable Amount: ",Total_Price-discount)
```

```
Discount Rupess: 54.2
Net payable Amount:  487.8
```

Exercise

[ ]:
```

```

1.1.3 4.1.3 If-elif-else statement

- The `elif` statement is used to check for the multiple `True` expressions and so on if the condition is `True` it execute a block of code.
- Similar `elif` statement is consider to be an optional like `else` statement.
- Though, we can use `elif` statement multiple times in a block of code, but `else` statement can be used only ones

Examples

[ ]:
```
'''Its very good example to get
the Net payable amount by setting up some
discount at specific price range by using if-elif-else statement'''

Total_Price=int(input("Price: "))
if Total_Price<100:
    discount=Total_Price*0.04
    print ("Discount Rupess: "+str(discount))
elif Total_Price<500:
    discount=Total_Price*0.08
    print ("Discount Rupess: "+str(discount))
```

```
else:
    discount=Total_Price*0.1
    print ("Discount Rupess: "+str(discount))
print ("Net payable Amount: ",Total_Price-discount)
```

```
Discount Rupess: 33.68
Net payable Amount:  387.32
```

Exercise

[ ]:

1.1.4 4.1.4 Nested If statement

- In some of the cases, it is required to have other check condition inside a check condition (when it `True`)
- Such type of scenarios required to have a nested arrangement.
- Under a nested if arrangement, an `if-elif-else` can be construct inside an another `if-elif-else` construct.

Examples

[ ]:
```
'''Its a simple example to check whether
a number is zero or positive or negative
using nested if-else statement'''

val = float(input("Enter a number: "))
if val >= 0:
    if val == 0:
        print("Zero")
    else:
        print("Entered number is Positive")
else:
    print("Entered number is Negative")
```

```
Entered number is Negative
```

Exercise

[ ]:

1.1.5 4.1.5 Single Statement Suites

- If the suite of `if` section have only a single line, then the header statement may go at same line.

Examples

[ ]:
```
'''Its a simple example to check whether a value is less than equal to 999'''
```

```
val = int(input("Enter a number: "))
if (val <= 999 ): print ("Value is less than equal to 999")
print ("Good bye!")
```

```
Value is less than equal to 999
Good bye!
```

Exercise

[ ]:

1.2 4.2 Loops Statements

- In loop, the statements are sequentially executed, execution of first function statement is done first, then second and so on.
- In situation where you need to perform a block of code many times then loop statement will come in to picture.
- It allows to execute/run group of statements or a statement multiple time.

These are several topics of loop statements which going to be discussed below: 1. **while Loop** 2. **for Loop** 3. **Nested loops** 4. **Loop Control** * **break** * **continue** * **pass**

1.2.1 4.2.1 While Loop

- It repeats a statement or multiple statements when a provided condition is `True`.
- Before performing a loop body it check for a condition

Examples

[ ]:
```
''' A child is learning to count number of 10 rupess in his hand
and here we have given 5 notes of 10 rupess.
Output will display the counting of notes and total amount'''

notes = 5
i=1 # starting number intialize
while i<=notes:
    print(i)
    i += 1
print('Total Money: ',notes*10,'Rupees')
```

```
1
2
3
4
5
Total Money:  50 Rupees
```

[ ]:
```
'''Continuing with same example,
here we using else statement'''
```

```
notes = 5
i=1 # starting number intialize
while i<=notes:
    print(i)
    i += 1
else:
    print("No 10 rupees note left")
print('Total Money: ',notes*10,'Rupees')
```

```
1
2
3
4
5
No 10 rupees note left
Total Money:  50 Rupees
```

Exercise

[ ]:

1.2.2 4.2.2 For Loop

- Here `for` loop statement is used to iterates over an item at any order, items can be a `string` or a `list`.

Examples

[ ]:
```
''' A simple example has taken where
we have several fruits in a list and we display
this fruit as per the arrangement in list'''

fruits_list = ["Mango","Cherry","Apple","Papaya","Banana"] ## List containing␣
 ↪friuts name
for x in fruits_list:
    print(x)
```

```
Mango
Cherry
Apple
Papaya
Banana
```

[ ]:
```
'''Continuing the previous example with using else statement'''

fruits_list = ["Mango","Cherry","Apple","Papaya","Banana"] ## List containing␣
 ↪friuts name
for x in fruits_list:
    print(x)
```

```
else:
    print("List has no fruit left!")
```

```
Mango
Cherry
Apple
Papaya
Banana
List has no fruit left!
```

[ ]:
```
'''An iterative example of string is shown here'''

f_name = "Mango" ## string as input
for x in f_name:
    print(x)
```

```
M
a
n
g
o
```

Exercise

[ ]:
```

```

1.2.3 4.2.3 Nested Loops

- Nested loops is used to create one and more number of loop inside an existing `for` or `while` loop.

Examples

[ ]:
```
'''Here we use nested for loop to create a triangle'''

n=7 #n is length and width of triangle
for i in range(0, n): #loop to handle the rows

    for j in range(0, i+1): #loop to handle the columns

        print("* ",end="") #printing the stars

    print("\r") #after each row line will end
```

```
*
* *
* * *
* * * *
* * * * *
```

```
* * * * * *
* * * * * * *
```

[ ]:
```
''' This exaple shows the use of nested while loop to
create a pyramid like structure'''

num_rows = int(input("Enter the number of rows: "))
row = 0 #row intialize
while(row < num_rows):
    row += 1 #Rows count increase
    s = num_rows - row #Spaces

    sc = 0 #Space counter intialize
    while(sc < s):
        print(" ", end='')
        sc += 1

    stars = 2*row-1 #Number of stars
    while(stars > 0):
        print("*", end='')
        stars -= 1

    print()
```

```
    *
   ***
  *****
 *******
*********
```

Exercise

[ ]:
```

```

1.2.4 4.2.4 Loop Control

The loop control is used to change its preceding sequence to different form.

break statement - This statement used for a premature terminates of current loop immediately and executes the further statement, just like `break` statement in `C` language.

continue statement - When `continue` statement is encountered during an iteration, it directly goes to next iteration without performing remaining statements in present iteration. It returns execution to the starting point of current loop and start further iteration.

> **Note:** `break` and `continue` can be used in both `while` and `for` loop.

pass statement - The `pass` statement is considered as a `null` operation. It is useful when a code (i.e., code required in future) has not written till now, but your code will go eventually.

Break statement Examples

[ ]:
```
'''It is a simple example of break,
where it immediatly terminate an operation of
statement body when x is equal to 4'''

x = 1
while x < 7:
    print(x)
    if x == 4:
        break
    x += 1
```

```
1
2
3
4
```

[ ]:
```
'''Here it terminates the
operational body when fruit name is Papaya'''

fruits_list = ["Mango","Cherry","Apple","Papaya","Banana"] ## List containing␣
 ↪friuts name
for x in fruits_list:
    if x == "Papaya":
        break
    print(x)
```

```
Mango
Cherry
Apple
```

Exercise

[ ]:
```

```

Continue statement Examples

[ ]:
```
'''It is a simple example of continue,
where it simply terminate an current iteration
when x is equal to 4 and
continues with next iteration'''

x = 0
while x < 7:
    x += 1
    if x == 4:
        continue
```

```
    print(x)
```

```
1
2
3
5
6
7
```

[ ]:
```
'''In this example it simply terminate
an current iteration when fruit name is Papaya and
continues with next iteration'''

fruits_list = ["Mango","Cherry","Apple","Papaya","Banana"] ## List containing␣
 ↪friuts name
for x in fruits_list:
    if x == "Papaya":
        continue
    print(x)
```

```
Mango
Cherry
Apple
Banana
```

Exercise

[ ]:
```

```

Pass statement Examples

[ ]:
```
'''A simple example of pass,
where it is not decided what to do with i'''

for i in [10, 100, 200]:
    pass
```

[ ]:
```
'''Continuing with same above example
if for loop body left simply, then it will throw an error'''

for i in [10, 100, 200]:
```

```
  File "<ipython-input-35-04f54b39379d>", line 1
    for i in [10, 100, 200]:
                           ^
SyntaxError: unexpected EOF while parsing
```

Exercise

[ ]:

[ ]: