[3.1] Iteration

January 5, 2023

1 Loops

In loops, based on the mentioned/given condition, the statement(s) perform action(s) over and over, while and for are two main loops in Python for this purpose.

1.1 while loop

while loop continually perform an action, until some condition has been met. Before executing the statement(s), that are nested in the body of the loop, Python keeps evaluating the test expression (loop test), until the test returns a False value.

Let's start with a simple while loop in the cell below.

[1]:
```
# A simple while loop
i = 1 # initializing a variable i = 1
while i < 5: # loop test
    # Run the block of code (given below), till "i < 5"
    print('The value of i is: {}'.format(i))
    i = i+1 # increase i by one for each iteration
```

```
The value of i is: 1
The value of i is: 2
The value of i is: 3
The value of i is: 4
```

What if we don't have i = i+1 statement? Well, we will get into an infinite while loop, because i will alway be less than 5, right? This is not what we want…….!

Note: If you get into an infinite while loop, you will notice a continuous out put or asterisk on left side of your cell in the jupyter notebook for a very long time **"In [*]"** . Go to the Kernel and restart to get out of this situation

A nice and cleaner way of exiting the while loop is using else statement (optional):

[2]:
```
i = 1
while i < 5:
    print('The value of i is: {}'.format(i))
    i = i+1
else:
    print('Exit loop')
```

```
The value of i is: 1
The value of i is: 2
The value of i is: 3
The value of i is: 4
Exit loop
```

1.2 for loop

"for" loop is a common loop that we will be using most of the time. This loop allows us to iterate through a sequence. The for statement works on strings, lists, tuples and other built-in iterables, as well as on new user-defined objects.

Let's create a list "my_list" of numbers and run a "for" loop using that list.

[3]:
```
my_list = [1,2,3,4,5]
```

In a very simple situation, we can execute some block of code using "for" loop for every single item in "my_list", let's try this out!

[4]:
```
for item in my_list:
    print(item)
```

```
1
2
3
4
5
```

Note: In the "for" loop above, the temporary variable "item" can be whatever you want e.g. i, a, x, name, num etc. A good practice is to use it carefully so that you can be self-explanatory and easier to remember. A good selection in this case is "num", because we have numbers in my_list.

TIP: A very useful trick is, always use comments in your code, this is a great way to remember and also helpful while working in a team.

[5]:
```
for num in my_list:
    print('Hello world')
```

```
Hello world
Hello world
Hello world
Hello world
Hello world
```

Did you notice something in the above cell? I have replaced "item" with "num" and also the block of code in the print statement. So, the print('Hello world') does not need to be related to items in the sequence (my_list in this case), we can print anything, we want!

Let's print the square of the numbers in my_list using for loop in **printSquare** function

[6]:
```
def printSquare(myList):
    # printing square of the numbers in myList.
```

```
    for num in myList:
        print(num**2)
```

[7]:
```
#calling printSquare function
printSquare(my_list)
```

```
1
4
9
16
25
```

This was all about while and for loops at the moment!

A quick reminder: While working in jupyter notebook, <shift + tab> is always useful to expore the document string of the related method.

After discussing loops, its time to introduce a very useful loop-related function range( ).

range():

Rather than creating a sequence (specially in for loops), we can use range(), which is a generator of numerical values. * With one argument, range generates a list of integers from zero up to, but not including, the argument's value. * With two argument, the first is taken as the lower bound (start). * We can give a third argument as a step, which is optional. If the third argument is provided,Python adds the step to each successive integer in result (default value of step is "+1").

Some working examples of range are given in the below code cels.

[8]:
```
# This (the code below) will give the range object, which is iterable.
range(5)
```

[8]:
```
range(0, 5)
```

[9]:
```
# With method "list", we can convert the range object into a list
list(range(5))
```

[9]:
```
[0, 1, 2, 3, 4]
```

[10]:
```
# Use of range(), with single argument, in a loop
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

[11]:
```
# Use of range(), with two argument, in a loop
for i in range(3,5):
```

```
    print(i)
```

```
3
4
```

[12]:
```
# Use of range(), with three argument, in a loop
for i in range(1,10,2):
    print(i)
```

```
1
3
5
7
9
```

Let's implement a common loop algorithm Sum of Values

[13]:
```
"""
Sum of Values
    • Initalize total to 0
    • Use while loop with sentinel

"""
def sum_of_values():

    total = 0.0
    inputStr = input("Enter value: ")

    while inputStr != "" :
        value = float(inputStr)
        total = total + value
        inputStr = input("Enter value: ")
    print("Sum of values :",total)
```

[14]:
```
#calling sum_of_values function
sum_of_values()
```

```
Enter value: 10
Enter value: 25
Enter value: 30
Enter value:
Sum of values : 65.0
```

Comparing Adjacent Values

- Get first input value
- Use **while** loop to determine if there are more to check
  - Copy input to previous variable
  - Get next value into input variable

– Compare input to previous, and output if same

[15]:
```
def compare_adjacent():
    value = int(input("Enter a value: "))
    inputStr = input("Enter a value: ")

    while inputStr != "" :
        previous = value
        value = int(inputStr)

        if value == previous :
            print("Duplicate input")
        inputStr = input("Enter a value: ")
```

[16]:
```
#calling compare_adjacent function
compare_adjacent()
```

```
Enter a value: 10
Enter a value: 15
Enter a value: 10
Enter a value: 10
Duplicate input
Enter a value:
```

2 Exercise

2.1 Budget Analysis

Write a program that asks the user to enter the amount that he or she has budgeted for a month. A loop should then prompt the user to enter each of his or her expenses for the month and keep a running total. When the loop finishes, the program should display the amount that the user is over or under budget.

2.2 Average Rainfall

Write a program that uses nested loops to collect data and calculate the average rainfall over a period of years. The program should first ask for the number of years. The outer loop will iterate once for each year. The inner loop will iterate twelve times, once for each month. Each iteration of the inner loop will ask the user for the inches of rainfall for that month. After all iterations, the program should display the number of months, the total inches of rainfall, and the average rainfall per month for the entire period.

2.3 Pennies for Pay

Write a program that calculates the amount of money a person would earn over a period of time if his or her salary is one penny the first day, two pennies the second day, and continues to double each day. The program should ask the user for the number of days. Display a table showing what the salary was for each day, and then show the total pay at the end of the period. The output should be displayed in a dollar amount, not the number of pennies.

2.4 Ocean Level

Assuming the ocean's level is currently rising at about 1.6 millimeters per year, create an application that displays the number of millimeters that the ocean will have risen each year for the next 25 years.

2.5 Guess the Number

Write a script that plays "guess the number." Choose the number to be guessed by selecting a random integer in the range 1 to 1000. Do not reveal this number to the user. **Display the prompt "Guess my number between 1 and 1000 with the fewest guesses:"**. The player inputs a first guess. If the guess is incorrect, display **"Too high. Try again."** or **"Too low. Try again."** as appropriate to help the player "zero in" on the correct answer, then prompt the user for the next guess. When the user enters the correct answer, display **"Congratulations. You guessed the number!"**, and allow the user to choose whether to play again.

2.6 Population

Write a program that predicts the approximate size of a population of organisms. The application should allow the user to enter the starting number of organisms, the average daily population increase (as a percentage), and the number of days the organisms will be left to multiply. For example, assume the user enters the following values:

Starting number of organisms: 2

Average daily increase: 30%

Number of days to multiply: 10

The program should display the following table of data:

Day	Approximate Population
1	2
2	2.6
3	3.38
4	4.394
5	5.7122
6	7.42586
7	9.653619
8	12.5497
9	16.31462
10	21.209

[ ]: