High Impact Skills Development Program

AI & Data Science

Lab 03: Loops in Python

Lab 03: Data Structures & Loops in Python

Introduction
The purpose of this lab is to get familiar with usage of while loop and 'for' loop in Python.

Objectives
The objective of this lab is to design solutions using a while loop or 'for' loop in Python.

Tools/Software Requirement
Jupyter Notebook

Description:

There are 4 built-in data types in Python used to store collections of data, these are List, Tuple, Set, and Dictionary, all with different qualities and usage.

List:

```
thislist = ["apple", "banana", "cherry"]

print(thislist)
```

- Lists are used to store multiple items in a single variable ,
- List items are ordered, changeable, and allow duplicate values.
- List items are indexed, the first item has index `[0]`, the second item has index `[1]` etc.
- The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.
- Since lists are indexed, lists can have items with the same value.
- To access a value in the list we can use ,
    - thislist[1]
    - thislist[-1]
    - thislist[2 : 5]
    - thislist[: 4]
    - thislist[2 :]
- append( ) and insert ( ) are used to add a new item to the list
    - thislist.**append**("orange") [Adds at the end of the list]
    - thislist.**insert**(1, "orange") [Adds at the specific index]
- remove( ) and pop ( ) are used to remove an item from a list.
    - thislist.remove("banana")
    - thislist.pop(1)
    - thislist.pop( )

Tuple:

thistuple = ("a", "b", "c")

print(thistuple)

- A tuple is a collection which is ordered and unchangeable.
- Tuples are used to store multiple items in a single variable.
- Tuple items are ordered, unchangeable, and allow duplicate values.
- Tuple items are indexed, the first item has index `[0]`, the second item has index `[1]` etc.
- When we say that tuples are ordered, it means that the items have a defined order, and that order will not change.
- Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.
- Since tuples are indexed, they can have items with the same value:
- To access a tuple item we can use its index,
-
    - thistuple[1]
    - thistuple[2:5]
    - thistuple[-1]

Set:

myset = {"apple", "banana", "cherry"}

- Sets are used to store multiple items in a single variable.
- A set is a collection which is **unordered**, **unchangeable***, and **unindexed**.
- Set items are unordered, unchangeable, and do not allow duplicate values.
- Unordered means that the items in a set do not have a defined order.
- Set items can appear in a different order every time you use them, and cannot be referred to by index or key.
- Set items are unchangeable, meaning that we cannot change the items after the set has been created.
- Sets cannot have two items with the same value.
- You cannot access items in a set by referring to an index or a key.But you can loop through the set items using a `for` loop, or ask if a specified value is present in a set, by using the `in` keyword.

```
thisset = {"apple", "banana", "cherry"}

for x in thisset:

    print(x)
```

- To Remove an item from a set,
    - thisset.remove("banana")
- To Add an item to a set,
    - thisset.add("orange")

Dictionary:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection which is ordered*, changeable and do not allow duplicates.
- Dictionary items are ordered, changeable, and does not allow duplicates.
- Dictionary items are presented in key:value pairs, and can be referred to by using the key name.
- Dictionaries are changeable, meaning that we can change, add or remove items after the dictionary has been created.
- Dictionaries cannot have two items with the same key:
- To add an item into a dictionary we use,
    - thisdict["color"] = "red"
- To remove an item from a dictionary we use,
    - thisdict.**pop**("model")
- To access a certain value in a dictionary we use
    - x = thisdict.**get**("model")

While Loop:

While loop is the simplest loop, which executes one or more statements, if the given condition remains true. It is useful when the number of iterations is not known in advance.

```
while «expression»:
      «block»
```

Example:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

For Loop:

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

```
for «val» in «sequence»:
      «statement»
```

Example:

```
languages = ['Swift', 'Python', 'Go', 'JavaScript']

for language in languages:

    print(language)
```

Lab Tasks

Using only the programming techniques that you have learned so far, perform the following tasks:

Task 1: Write a program that asks the user about the number of values he/she wants to enter. Than enter the values as per the required number, calculate its sum and identify the smallest value among them. The sample output is as follow:

```
Enter the number of values to be input: 5
Enter the number: 20
Enter the number: 10
Enter the number: 50
Enter the number: 4
Enter the number: 65
The sum is: 149
The smallest value of entered numbers is 4
```

Task 2: The factorial function is used frequently in probability problems. The factorial of a positive integer n (written n! and pronounced "n factorial") is equal to the product of the positive integers from 1 to n.
Write a function factorial that accepts an integer as parameter and returns its factorial.
Using the factorial function, write a program that evaluates the factorials of the integers from 1 to 5. Print the results in tabular format as following.

```
X       Factorial of X
1       1
2       2
3       6
4       24
5       120
```

Task 3: Write a program that plays an incredibly stupid number-guessing game. The user will try to guess the secret number until they get it right. That means it will keep looping as long as the guess is different from the secret number. You must store the secret number in a variable, and use that variable throughout. The secret number itself must not appear in the program at all, except in the one line where you store it into a variable. Sample output is as following:

```
I have chosen a number between 1 and 10. Try to guess it.
Your guess: 5
That is incorrect. Guess again.
Your guess: 4
That is incorrect. Guess again.
```

```
Your guess: 8
That is incorrect. Guess again.
Your guess: 6
That's right! You guessed it.
```

Task 4: The greatest common divisor (GCD) of two integers is the largest integer that evenly divides each of the two numbers. Write function gcd that returns the greatest common divisor of two integers.
Use the gcd function in your program to determine the GCD of the numbers in the sample output:

```
Enter two integers: 75 225
The greatest common divisor of 75 and 225 is 75

Enter two integers: 99 30
The greatest common divisor of 99 and 30 is 3

Enter two integers: 17 22
The greatest common divisor of 17 and 22 is 1

Enter two integers: 100 92
The greatest common divisor of 100 and 92 is 4

Enter two integers: 10005 15
The greatest common divisor of 10005 and 15 is 15
```

Task 5: An integer is said to be prime if it is divisible only by 1 and itself. For example, 2, 3, 5 and 7 are prime, but 4, 6, 8 and 9 are not.
a) Write a function that determines if a number is prime.
b) Use this function in a program that determines and prints all the prime numbers between 1 and 10,000.

```
The prime numbers from 1 to 10000 are:
     1     2     3     5     7    11    13    17    19    23
    29    31    37    41    43    47    53    59    61    67
    71    73    79    83    89    97   101   103   107   109
   113   127   131   137   139   149   151   157   163   167
    .
    .
    .
  9733  9739  9743  9749  9767  9769  9781  9787  9791  9803
  9811  9817  9829  9833  9839  9851  9857  9859  9871  9883
  9887  9901  9907  9923  9929  9931  9941  9949  9967  9973
```

Task 6: Write a code that prints on screen all the 4-digit Armstrong numbers.

Task 7: Write a program that outputs 100 lines, numbered 1 to 100, each with your name on it. The output should look like the output below:

```
1 Your name
2 Your name
3 Your name
4 Your name
...
100 Your name
```

Task 8: Write a program that prints out a list of the integers from 1 to 20 and their squares. The output should look like this:

```
1 --- 1
2 --- 4
3 --- 9
...
20 --- 400
```

Task 9: Write a program that uses a for loop to print the numbers 8, 11, 14, 17, 20, . . . , 83, 86, 89.

Task 10: Write a program that asks the user for their name and how many times to print it. The program should print out the user's name the specified number of times.

Task 11: Use a for loop to print a box like the one below. Allow the user to specify how wide and how high the box should be.

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

Task 12: Use a for loop to print a triangle like the one below. Allow the user to specify how high the triangle should be.

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

Task 13: Use for loops to print a diamond like the one below. Allow the user to specify how high the diamond should be.

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

Task 14: Write a program that prints a giant letter A like the one below. Allow the user to specify how large the letter should be.

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