Introductory Python Course - Session 2

Delivered to you by DUCSS

Last week we covered

- What and why: python
- VS code & python interpreter
- A few things about variables, operations on them and user input/output

New material - Theory

- Beginners (correspond to videos 19 - 28 in Microsoft Python Course)
  - Conditional execution (if elif else statements)
  - Repeated execution (for, while)
  - Arrays (objects containing multiple items of the same type)
  - Operations on arrays
- Advanced (If you are a beginner do not worry about this at all):
  - Sorting algorithms and binary search and their applications in interview questions

Exercise - 1 Solution

- Create a program which asks for the length and width of a rectangle from the user and prints its area and perimeter.
- https://github.com/DUCSS/Python-Workshops/blob/main/week%201/exercise-1.py

Homework / Exercise 2 Solution

- Learn how to do division in Python and hence come up with a program that takes in 4 decimal values and prints their mean
  - For that you will have to search something about “integer division” and “floating point division” and understand the differences between the two.
- https://github.com/DUCSS/Python-Workshops/blob/main/week%201/homework.py

Typical conditional execution structure

```
if (condition):          ->  one “if” statement (executed if “condition” is satisfied)
    statements
elif (condition):        ->  zero or more “else if” statements
    statements
else:                    ->  zero or 1 else statement (executed if all conditions fail)
    statements
```

NB: only one of the branches is executed within one sequence of conditional statements be it: if, elif or else

Conditions

- Anything that can be evaluated to True/False. Also called “booleans” as a memory of Geogre Boole who worked on algebraic logic
- a > b, a<=b, a>=b, a==b (equality), a != (inequality) -> common mathematical statements
- For example, 10 < 1 evaluates to False, 10 > 1 evaluates to True, 10 == 10 evaluates to True, 5 >=5 evaluates to True, 5 !=4 evaluates to True, 2!=2 evaluates to False
- We can also use variables, as shown above i.e a==b

Complex conditions

- Suppose we want to check an integer lies in interval [1, 5] (1<=number<=5)
- We can encode it as:
  - number >=1 and number <=5
- We can use complex conditions in conditional execution:
  - if number >=1 and number <=5:

    print(“number lies in the interval”)

- We can use the following clauses to combine the conditions: and, or. We can also negate the condition by saying not(condition).

Example of conditional execution - Code Link

- Suppose you are working for an amusement park company and you need to write a program that checks that the client's age is greater than 14 to make sure that it will be safe for them to be on a particular attraction.
- Write a program that outputs "YES" if the client's age is greater than 14 and "NO" if the client's age is less than or equal to 14.
- If the age entered is invalid, i.e the number is less than zero notify the user about by printing "ERROR"

Repeated execution

- Sometimes we may want to do a particular set of statements a number of times.
- A simple solution would be copying and pasting these statements one after the other many times, but what if we want the user to determine the number of times something is executed?
- We use while and for loops for that.

While loop structure

- while condition:

    statements

- This means execute “statements” while the condition remains true.
- If the condition is False initially the while loop is skipped
- Normally the condition would be changed by the statements, so the loop should stop at some point.
- If the loop never ends it is called “infinite” it is equivalent to saying:
- while True:

    statements

Example of while loop - Code Link

- Suppose you want to print all powers of 2 which are less than a particular number that the user inputs
- A power of 2 is a number of the form 2^k where k is positive integer or zero, for example 2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 =16 …

Continue and break statements - Code Link

- Each execution cycle of a loop is called iteration
- “continue” statement makes the loop immediately proceed to the next iteration
- “break” statement makes the loop exit immediately
- Using these statements is considered a BAD PRACTICE, they are covered so that you know they exist, never use them in code you write (unless you really have to)

Arrays/Lists - Collections of items of the same type - Code Link

- Arrays are defined as array_name = [elem1, elem2 ...]. The list of elements maybe empty [ ] and may contain variables and constants [10, a, 11, b ...]
- If we want to access an element at a particular index we do array_element[index]
- If we want to add an element to the end of the array we do array_name.append(element)
- If we want to add multiple elements to the end of the array we do array_name.extend(another_array_of_elements)
- There are other inbuilt methods such as remove and etc.

for/foreach loop structure - Code Link

- for index in range(lower_bound, upper_bound, step):

  statements

- for element in array:

  statements

- range(lower_bound, upper_bound, step) returns an array which starts with lower_bound and where each of the next elements is greater than the previous one by step. The last element has to be smaller than upper_bound

Register on Hackerrank

- https://www.hackerrank.com/ - choose “Sign up and Code”
- Questions we will be covering for beginners cohort during the next session:
  - https://www.hackerrank.com/challenges/python-arithmetic-operators/problem?isFullScreen=true
  - https://www.hackerrank.com/challenges/py-if-else/problem?isFullScreen=true
  - https://www.hackerrank.com/challenges/find-second-maximum-number-in-a-list/problem?isFullScreen=true
  - https://www.hackerrank.com/challenges/array-left-rotation/problem?isFullScreen=true

Register on Hackerrank

- Questions for advanced cohort:
  - https://www.hackerrank.com/challenges/ctci-bubble-sort/problem?isFullScreen=true&h_l=interview&playlist_slugs%5B%5D=interview-preparation-kit&playlist_slugs%5B%5D=sorting
  - https://www.hackerrank.com/challenges/fraudulent-activity-notifications/problem?isFullScreen=true&h_l=interview&playlist_slugs%5B%5D=interview-preparation-kit&playlist_slugs%5B%5D=sorting

Thank you!