Python Programming Workshop Session 3

Avnith Vijayram
https://avnithv.github.io/

Table of Contents

Review of For loops

Used for iterating through a sequence

The sequence can be strings, lists, and the range() function

The program assigns the current item in the sequence to a variable

The code inside is executed as many times as there are items in the sequence.

This is the most common and useful control structure

Review of lists

Lists are ordered and mutable sequences of values

Each item in a list has its own index, which is used to access it.

Lists can be changed and still be assigned to the same variable.

They are similar to strings in the way that they can be indexed and sliced

append() and insert() methods are used to add items, del and pop() are used to delete

Review of dictionaries

Dictionaries are an unordered and mutable set of key-value pairs

Each key corresponds to a value, and each value is accessed through its key

The same key cannot have multiple values, but multiple keys can have the same value.

Dictionaries do not have indices because they are not ordered, and they can be changed.

Dict[key] = value and update() add key-value pairs

Del and pop() remove key value pairs

Review of Functions

Functions are blocks of code that are written as one line.

They can take inputs and give outputs.

Parameters can have default values

Arguments can be passed positionally or through keywords

They make code more clean and efficient.

How to write a program

There are multiple steps when writing a program.

First, understand the problem and then write how you would solve it.

Write a pseudocode and create an outline

Write the code with syntax rules and details

Use error messages to find any mistakes in the code.

Step 1: Solve the problem

Read through the whole problem

Understand how to solve the problem

Try solving the sample problems without looking

One of the biggest mistakes when writing a program is not fully knowing what you need to solve.

Step 2: Write a pseudocode

After solving the problem, write down the steps you took to solve the problem.

These steps don’t have to be detailed and precise.

This is called writing pseudocode.

Pseudocode is a plain language description of the steps of an algorithm or program.

Pseudocode includes programming conventions but is meant for human reading.

Step 3: Create an outline

Define the main functions and control structures that you will use.

This will help you keep your code organized and know what to write.

Include the parameters and returned values when you outline a function.

Don’t include any of the code inside yet.

Step 4: Fill in the details

Fill in the details of the functions and the control structures in the outline. This includes:

Handling the input and output

Writing the functions that each solve a smaller problem

Calling the functions that you created

Checking that the code is free of syntax errors.

Step 5: Check your code

Run the code with the sample inputs

If the code gives an error, trace the error message.

If the code gives the wrong answer, troubleshoot by printing values in certain places.

Create your own inputs and outputs and check those.

Find any weaknesses in your code

What are errors?

Errors are problems in the code where the computer does not understand what to do

Python includes a traceback, line number, line code, error type, and description in the error message.

SyntaxError - Missing quotes on a string, missing a colon after a control structure, missing a parenthesis.

TypeError - Using an operator on the wrong type, calling a function with missing or wrong type arguments

IndentationError - Missed or added an extra indent on a control structure.

NameError - Not defined a variable, misspelt a variable, not put quotes on a one word string.

How to use errors

Errors are useful when you are trying to fix your code.

Go to the line where the error is occurring. Unless it is a syntax error, the actual error will be near that line.

Use the description in the error message to find what exactly is going on.

Fixing errors in your code may take the same amount of time as writing it, and this is okay.

Troubleshooting

If the code does not raise an error but gives the wrong answer, it might be useful to troubleshoot it.

Troubleshooting helps you trace where the mistake is happening.

Print the values of important variables in important places in the code, such as within a for loop or a function.

If at any point, the value of a variable is not what it should be, then you know the mistake occurred before this and after the last time it was printed correctly

With that information, you should be able to find the mistake or narrow it down further..

Finding potential errors

Even if your code works perfectly for the sample inputs, it may not work for the trickier inputs of the actual test.

Your code may be based on assumptions of what the input will be, or may overlook a possibility.

It is important to read the restrictions for the input because that will determines all of the possible inputs.

For example, if the definition of a variable is based on a condition being true or a control structure being executed, if that is not satisfied, the variable will not exist and Python will raise a NameError