Introduction to Programming Using Python – Homework Exercises

Introduction

These homework exercises are to help the student judge their progress in the subject. Do not fret if you have difficulties. Do not assume if they are easy that you shall not learn some interesting and useful items.

1. **Please remember**

* Homework is for your benefit / fun.
* Do only as much / or as little as you can.
* Contact me with problems/successes/questions.
* Have fun!

1. **If you have problems:**

* Think how you might solve the problem by hand.
* Try simplifying the task, in hope of adding to your solution as time permits.
* Ask the instructor for help. email?

1. **If the solution appears too easy:**

* Consider if you have completely solved the problem. Error checking?
* Have you thoroughly tested your solution?
* Have you automated (written a program) to validate your solution?

1. **Please feel encouraged to email your solution and any other comments or questions to the instructor.**
2. **Name you program files with short descriptive names e.g., hello\_world.py. All lowercase names are usually easiest to type and remember. Please test run your program(s), storing the output(s) in """ strings at the bottom of the source file or separate files named *program\_file\_without\_extension*.out *e.g,* hello\_world.out for program file named hello\_world.py.**

Running and saving your program's output can be easily done by:

1. Starting IDLE, if not running
2. Open you program, if not present (**File**🡪**Recent** **Files**🡪*your file*
3. Running you program till done (**Run**🡪**e.g., Module** **(F5)**)
4. Save IDLE shell (File🡪**Save As**

**Save as Type:** All Types (\*.\*)

**File:name:** *program\_name***.out**

**SAVE**

)

Since you saved file is the most recently saved text file, you can look and edit this output by doing (**File**🡪**Recent Files->***your file***)**

1. **Our solution files / alternatives are mostly in folders …Introduction…/homework/Class\_.../**

Our solutions for our twenty-question project are in …Introduction…/exercises/twenty\_question\_dev/iteration\_1.py, iteration\_2.py…

Please be encouraged to look at them if:

* You get stuck (Your always welcome to ask me for help via email/class)
* You have completed your solution and want to compare
* Are otherwise curious

If time permits, one gets the most out trying their own solution first.

Class 1: Introduction - Getting Going

1. **Everything about Python we saw today**

Make a list of all the topics we covered today. You don't need to be an expert in the topic. You need not be complete. A prize to the one who has the most items.

A start:

|  |  |  |
| --- | --- | --- |
| Topic | Example | Use |
| A program is set of instructions | print("Hello World") | Task process |
| Arithmetic operators | print(1+2-3\*4/5) | Calculation |
| Variables | a, max, customer\_name | Store values for later use |
|  |  |  |

1. **Hello world program (Our solutions: exerceses/introduction/hello\_world.py,…)**

* Create and run a new file **hello\_world.py** - nothing new – just practice.
* Create and run a slightly different file **goodbye\_world.py**, using your first program as a starting point.

1. **Everything Program**

Create a program file named "everything\_python\_1.py" which contains at least one example of every item you listed in 1. If any of your items intentionally cause the program to stop, write each of these as separate program files named "everything\_1\_err\_type" where type is tag describing the error type, e.g., "syntax" for syntax error.

You may combine items within one statement, but please indicate or comment items when used first time. Prizes given for: most items used, smallest program with most items

1. **Twenty questions Class Project Progress**

It's a marathon (short 😊), not a sprint.

1. Review / redo iteration 1 we did in class.

Iteration 1 – Loop forever, prompting for guess, accepting guess, printing guess

Write and test Iteration 2. Remember start by saving a copy of iteration1.py as a new file named iteration2.py.

Iteration 2 – ADDING to Iteration 1 code: Set target value, Quit loop if number entered number equals target

1. **MY\_ATM – the beginnings of an ATM machine program**

An ATM or Automatic Teller Machine facilitates the depositing and dispersal of money. Even without the mechanical aspects the programming involved can be very complex. Our program here will begin with just concerning the accepting and dispersing money. We will just consider the single individual having already logged on and verified. To further simplify the demands on the program only deposits or withdrawals will be accepted.

Program: File name: my\_atm.py

Operation:

1. Initial balance is 0.
2. Pressing the ENTER key with no amount will exit and display the balance
3. Positive number will withdraw that amount, IF sufficient balance, and display the new balance
4. Negative number will deposit that amount, and display the new balance.

Example **my\_atm.py** output:

Withdrawal AMT:50

Insufficient Funds for withdrawal

Balance: 0.0 request: 50.0

Withdrawal AMT:-100

Deposit: 100.0

New Balance: 100.0

Withdrawal AMT:75

Withdrawal: 75.0

New balance: 25.0

Withdrawal AMT:5

Withdrawal: 5.0

New balance: 20.0

Withdrawal AMT:-10

Deposit: 10.0

New Balance: 30.0

Withdrawal AMT:

Bye - Have a good day