Carleton University
Department of Systems and Computer Engineering
SYSC 1005 - Introduction to Software Development - Fall 2017

Lab 2 - Working with Variables, Visualizing Code Execution

Objectives

- To gain experience translating mathematical formulae into Python statements that use variables.
- To gain experience using the online Python Tutor to visualize the execution of the code.

Attendance/Demo/Grading

When you have finished all the exercises, call a TA, who will review your solutions to the exercises, ask you to demonstrate some of your solutions, and assign a grade. For those who don't finish early, a TA will review your solutions to the exercises you've completed, starting about 30 minutes before the end of the lab period. **Any unfinished exercises should be treated as "homework"; complete these on your own time, before your next lab.**

Getting Started

Step 1: Create a new folder named Lab 2 on the lab computer's hard disk (on the desktop or in your account's Documents folder) or in the M: drive on the network server.

Step 2: Launch Wing IDE 101. Check the message Python displays in the shell window, and verify that Wing is running Python version 3.6. If another version is running, ask a TA for help to reconfigure Wing to run Python 3.6.

Step 3: In the *Labs* section of the course cuLearn page, click the link Open Python Tutor in a new window. Verify that PyTutor is configured this way: "Write code in Python 3.6", "hide exited frames", "render all objects on the heap (Python/Java)" and "draw pointers as arrows". If necessary, select the correct options from the drop-down menus.

Any code we type in the Python shell in Wing 101 lost when we exit the program or restart the shell. Similarly, while the Python Tutor is a great tool for visualizing the execution of short pieces of code, it is not a complete program development environment, and it doesn't provide a way for us to save our code in a file. So, while you're working on the exercises, you'll use Wing's editor to prepare a file that contains a copy of your solutions.

Step 4: Click the New button in the menu bar. A new editor window, labelled untitled-1.py, will appear.

Step 5: From the menu bar, select File > Save As... A "Save Document As" dialogue box will appear. Navigate to your Lab 2 folder, then type lab2.py in the File name: box. Click the Save button. Wing will create a new file named lab2.py.

Exercise 1: To convert a temperature measured on the Celsius scale to the equivalent Fahrenheit temperature, we multiply the Celsius temperature by $\frac{9}{5}$, then add 32.

In Wing 101's Python shell, write two assignment statements that do the following:

1. Create a new variable named `degrees_c` and assign it the value `20.0`.
2. Convert the temperature bound to `degrees_c` to the equivalent temperature in degrees Fahrenheit. Assign the Fahrenheit temperature to a new variable named `degrees_f`.

Use the shell to verify that the value bound to `degrees_f` is `68.0`.

When your code is correct, copy/paste it from the shell to file lab2.py and save the file (click the Save button or select File > Save from the menu bar.)

Exercise 2: Copy/paste your solution to Exercise 1 into the Python Tutor editor.

Click the Visualize Execution button. Execute the code, one statement at a time, by clicking the Forward> button. Observe the memory diagram as the code is executed, step-by-step. Make sure you can answer these questions:

- When does variable `degrees_c` appear in the diagram?
- What is the name of the frame containing `degrees_c`?
- What does `degrees_c` contain?
- When does variable `degrees_f` appear in the diagram?
- What does `degrees_f` contain?

Exercise 3: In some countries, a vehicle's fuel efficiency is measured in miles per gallon. In other countries, the efficiency is measured in litres per 100 km. One Imperial gallon is equal to approximately 4.54609 litres. One mile is equal to approximately 1.60934 km.

In Wing 101's Python shell, write assignment statements that do the following:

1. Create a new variable named `mileage` and assign it the value `32` (which represents 32 miles per gallon).
2. Create new constants named `LITRES_PER_GALLON` and `KMS_PER_MILE` and binds these to the values `4.54609` and `1.60934`, respectively. (Note that the names of constant values are, by convention, usually written entirely in uppercase.)
3. Calculate the fuel consumption, measured in litres/100 km. Assign this value to a new variable named `fuel_consumption`.

Use the shell to verify that the value bound to `fuel_consumption` is approximately `8.83`.

When your code is correct, copy/paste it from the shell to file lab2.py and save the file.

Exercise 4: Click the link Open Python Tutor in a new window and configure PyTutor as described in *Getting Started*, Step 3. Copy/paste your solution to Exercise 3 into the PyTutor editor.

Click the Visualize Execution button. Execute the code, one statement at a time, by clicking the Forward> button. Observe the memory diagram as the code is executed, step-by-step. Make sure you can answer these questions:

- How many variables and objects are created when your solution to Exercise 3 is executed?
- What do the arrows in the memory diagram represent?

Exercise 5: Suppose you have some money (the *principal*) that is deposited in a bank account for a number of years and earns a fixed annual *rate* of interest. The interest is compounded *n* times per year.

The formula for determining the total amount of money you'll have is:

$$amount = principal\left(1 + \frac{rate}{n}\right)^{n \cdot time}$$

where:

- *amount* is the amount of money accumulated after *time* years, including interest.
- *principal* is the initial amount of money deposited in the account
- *rate* is the annual rate of interest, specified as a decimal; e.g, 5% is specified as 0.05
- *n* is the number of times the interest is compounded per year
- *time* is the number of years for which the principal is deposited.

For example, if $1,500.000 is deposited in an account paying an annual interest rate of 4.3%, compounded quarterly (4 times a year), the amount in the account after 6 years is:

$$amount = \$1,500\left(1 + \frac{0.043}{4}\right)^{4 \cdot 6} \approx \$1938.84$$

In the shell, write assignment statements that do the following:

1. Create new variables named `principal`, `rate`, `n` and `time`, and bind these variables to the values `1500`, `0.043`, `4` and `6` respectively.

2. Calculate the total amount of money in the account after `time` years. Assign this new value to a variable named `amount`.

Use the shell to verify that the value bound to `amount` is approximately `1938.84`.

When your code is correct, copy/paste it from the shell to file lab2.py and save the file.

Exercise 6: Click the link Open Python Tutor in a new window and configure PyTutor as described in *Getting Started*, Step 3. Copy/paste your solution to Exercise 5 into the PyTutor editor.

Click the Visualize Execution button. Execute the code, one statement at a time, by clicking the Forward> button. Observe the memory diagram as the code is executed, step-by-step.

Exercise 7: Review the description of *augmented assignment* on pages 21 and 22 of *Practical Programming*. Without using the Python shell or PyTutor to assist you, predict the value that will be bound to `x` after Python executes these two statements:

```
x = 3
x += x - x
```

Use the shell to check if your prediction was correct.

Exercise 8: Consider the following statements:

```
a = 9
b = 4
c = a * b
d = b
a = 2
b = 3
c = c + d
```

Without using the Python shell or PyTutor to assist you, predict the values that will be bound to `a`, `b`, `c` and `d` after Python executes these statements.

Click the link Open Python Tutor in a new window and configure PyTutor as described in *Getting Started*, Step 3. Type this code in the PyTutor editor (not the shell).

Click the Visualize Execution button. Execute the code, one statement at a time, by clicking the Forward> button. Observe the memory diagram as the code is executed, step-by-step.

Were your predictions correct?

Wrap-up

1. Remember to have a TA review your solutions to the exercises, assign a grade (Satisfactory, Marginal or Unsatisfactory) and have you initial the demo/grading sheet.

2. Remember to backup your project folder before you leave the lab; for example, copy it to a flash drive and/or a cloud-based file storage service.

Last edited: Sept. 23, 2017