CS-119 Lab #3

Student Learning Outcomes

- Creating a Python console application
- Declaring variables
- Assigning values to variables
- Arithmetic operations
- Selection structures and relational operators
- Sequential code
- Application testing; creating test data and verifying results

Overview

This lab provides an opportunity to apply the concepts and principles presented in chapter 3 of the PLD text.

To save time, space and reading, all the nitty-gritty details of how to create a Python code file in IDLE, how to create the pseudocode file, etc. will be mostly left out. Please refer to the chapter 1 lab if you need a more detailed refresher on the se steps. However, we will review problem analysis and pseudocode in quite a bit of detail in this exercise. We'll focus more on the chapter concepts and the Python code needed to complete this problem as well as things you'll need to know to complete the rest of this lab.

Exercise 1: A Guided Python Application – Portrait Studio

This exercise is based on exercise 9 on pages 112 and 113 of the text. In this exercise, a portrait studio has hired you to implement a charge calculator for taking portraits. Part of the charge is based on the number of subjects in the portrait.

Subjects in Portrait	Base Price
1	$100
2	$130
3	$150
4	$165
5	$175
6	$180
7 or more	$185

Table 3-4 Portrait prices

Portrait sittings on Saturday or Sunday cost 20 percent more than the base price.

Design a flowchart or pseudocode for a program that accepts the following data: the last name of the family sitting for the portrait, the number of subjects in the portrait, and the scheduled day of the week. Display all the input data as well as the calculated sitting fee.

Analyzing the Problem

From the previous chapters, we know we can break this problem down into input, process and output.

What inputs do we need? Let's look at the problem and see if we can identify them:

Portrait sittings on Saturday or Sunday cost 20 percent more than the base price.

Design a flowchart or pseudocode for a program that accepts the following data: the last name of the family sitting for the portrait, the number of subjects in the portrait, and the scheduled day of the week. Display all the input data as well as the calculated sitting fee.

Now that we've identified the inputs of last name, number of subjects and the scheduled day of the week, our pseudocode will look something like this:

```
Get last_name
Get num_subjects
```

Get day_of_week

Next, let's break down the processing.

In this exercise, last name really isn't used for anything. In a *real world* application, data such as last name, address, billing information, etc. would be entered and stored in either a file or database and used for billing, delivering the portraits, etc.

We've been given a fee schedule based on the number of subjects for calculating the base price.

Portraits sittings on Saturday or Sunday are 20% extra.

These are the 2 processing steps we need but is this clear enough that we could write code? Let's be honest with ourselves, probably not. So, let's break these steps down a little further. Keep in mind there are a *number of ways* this problem could be solved. What's presented here is simply one of a number of possible ways so don't take this as the only solution or the way you have to do it. If you have a different approach you want to take, go for it!

To calculate the base price using the table above, we need to code a decision structure. In the chapter, we saw that we can use an if/else (dual alternative) decision structure. Since we have a number of choices, we'll need to do an expanded, not covered in the PLD text version of this called an if/else if/else structure. Our pseudocode will look something like this:

```
if num_subjects = 1 then
        base_price = 100
else if num_subjects = 2 then
        base_price = 130
else if num_subjects = 3 then
        base_price = 150
else if num_subjects = 4 then
        base_price = 165
else if num_subjects = 5 then
        base_price = 175
else if num_subjects = 6 then
        base_price = 180
else  // default catch all 7 or more
        base_price = 185
end if
```

Decision structures like this are heavily used in real world programming so it's beneficial to get experience working with these.

The next thing is to determine if an additional 20% charge applies. This gets charged if the sitting is on Saturday or Sunday. Once again, there are multiple, possible ways to approach this. We could use an if/else if/else decision structure or we could use a single alternative structure and use the logical OR operator. Here's the pseudocode for both approaches. Both will work and there are other ways that would work too. The choice of which one to use here is really a matter of personal preference.

Dual alternative:

```
if day_of_week = 7 then
        base_price = base_price * 1.2
else if day_of_week = 1 then
        base_price = base_price * 1.2
else
        // do nothing – no additional charge
end if
```

Single alternative:

```
if day_of_week = 7 OR day_of_week = 1 then
        base_price = base_price * 1.2
end if
```

To help guide your final decision, which of these two structures looks easier to code later on?

Notice in both cases we simply accumulate or add to (well, multiply in this case) the base charge if the condition is true.

Another point here is the value 1.2 for the surcharge is often called a *magic number*; its meaning and purpose is unclear. This makes for code that is more difficult to read and understand. To make things more readable and maintainable, let's add a constant called SURCHARGE_PCT and assign it the value 1.2. In languages like Python, Java, C++ and C#, it is customary to declare constants in all upper case letters and use the underscore (_) to space multiple words. The idea is to make them really stick out and not be mistaken for variables. Since Python doesn't have constants and it turns out these are really variables, be careful not to change the values of your Python "constants" after you declare them.

```
SURCHARGE_PCT = 1.2
```

Revised Single alternative using constants:

```
if day_of_week = 7 OR day_of_week = 1 then
        base_price = base_price * SURCHARGE_PCT
end if
```

Last thing is output. Notice that our total price is actually stored in the variable baseCharge so the pseudocode is very simply the line:

```
Display base_price
```

Creating the Application

Create a file called *PortraitStudio* in IDLE. Be sure to create this on your USB drive if you are using the lab computers. Create a blank text file for your pseudocode and name it *pseudocode.txt*.

Complete the Pseudocode

Here's all the pseudocode we saw earlier all in one place. Keep in mind this sample pseudocode is a *suggested* approach that will work. You are welcome and encouraged to develop your own approach and style.

```
# portrait studio pseudocode
SURCHARGE_PCT = 1.2
SUNDAY = 1
SATURDAY = 7

# input
get last_name
get num_subjects
get day_of_week # assume day 1 = Sun, day 7 = Sat

# processing
# calc base price
if num_subjects = 1 then
    base_price = 100
else if num_subjects = 2 then
    base_price = 130
else if num_subjects = 3 then
    base_price = 150
else if num_subjects = 4 then
    base_price = 165
else if num_subjects = 5 then
    base_price = 175
else if num_subjects = 6 then
    base_price = 180
else # default catch-all 7 or more
    base_price = 185
end if

# calculate surcharge if applicable
if day_of_week = 1 or day_of_week = 7
    base_price = base_price * SURCHARGE_PCT
end if

# output
display base_price
```

Another thing we are going to do in this lab is generate some test data we can use when we test to ensure our application is working correctly and producing correct results. We need test data for a varying number of subjects (test the selection structure) and we also need to test the weekend surcharge feature. Having a good table of test input data and the expected results allows us to test our application and be able to tell at a glance if it is working correctly. Doing this in a spreadsheet or MS Word document table would be an excellent choice. To keep things simple, we will add it to our pseudocode. It doesn't need to be anything fancy but should have enough values that we can test all possible scenarios.

```
# portrait studio test data
Subjects    Day    Price
1           2      $100
3           4      $150
6           5      $180
8           7      $234 (test weekend surcharge)
```

Code the Application

Here's a screen shot of the Python code:

```
#!/user/bin/env python3

# Your Name Here!
# Cuyamaca College CS-119
# Lab 2, exercise 1, Portrait Studio


# import locale to do currency formatting
#export LANG=en_US.UTF-8
import locale
locale.setlocale(locale.LC_ALL, 'en-US') # use "en_US" on Mac

# declare variables & constants
SURCHARGE_PCT = 1.2
SUNDAY = 1
SATURDAY = 7
base_price = 0.0
day_of_week = 0
last_name = ""
num_subjects = 0

# input
last_name = input("Enter last name: ")
num_subjects = int(input("Enter number of subjects: "))
day_of_week = int(input("Day of week (1 = Sun, 2 = Mon, ... 7 = Sat: "))

# process - calculate base price based on number of subjects
if num_subjects == 1:  # don't forget the colon (:) at the end of each condition!
    base_price = 100   # indent each true block and the else block EXACTLY 4 spaces!
elif num_subjects == 2:
    base_price = 130
elif num_subjects == 3:
    base_price = 150
elif num_subjects == 4:
    base_price = 165
elif num_subjects == 5:
    base_price = 175
elif num_subjects == 6:
    base_price = 180
else:
    base_price = 185

# add surcharge for weekend sitting
if day_of_week == 1 or day_of_week == 7:
    base_price = base_price * SURCHARGE_PCT

# output
print("Last name: " + last_name)
print("Total price: " + locale.currency(base_price, grouping=True))
```

Be sure to test your application and verify everything works correctly.

Exercise 2 BMI Calculator

For this exercise, you will design and develop an application that allows the user to enter their height in inches, weight in pounds and then calculate their body mass index (BMI). To calculate BMI, we will use the following formulas:

$$bmi = kilograms / meters^2$$

Wait a minute! What's going on here? Before we can do the calculation above, we need to convert weight in pounds to weight in kilograms and height in inches to meters. We also need these formulas:

meters = inches / 39.36
kilograms = pounds / 2.2

Once you calculate the BMI percentage, you will need to display one of the messages below based on the BMI percentage calculated.

BMI range	Message
Less than 18.5	Under weight
18.5 – 24.99	Normal weight
25 – 29.99	Over weight
30 – 39.99	Obese
40 or over	Morbid Obesity

Ranges/messages are based on the BMI Calculator site

Be sure to complete pseudocode or a flowchart for this exercise ***and complete a table of test values that will test all the ranges given above***. Both of these deliverables are worth points! Also, test the application with your test data and verify it produces the results expected.

Exercise 3 Coyote Inn

Coyote Inn has hired you to develop a new application to estimate charges for staying at their new inn. Their nightly rates are based on months of the year and provided in the following table:

Months	Rate
1 – 3 (Jan – Mar)	$80/night
4 – 6 (Apr – Jun)	$90/night
7 – 9 (Jul – Sept)	$120/night
10 – 12 (Oct – Dec)	$100/night

To keep things simple, you can allow the month to be entered by the month number. Normally, this would be done by a date period but we'll do it by month to keep the exercise simple.

The user must be able to enter a month number (1 – 12) and the number of nights. Use a decision structure to find the correct price per night for the month entered and then multiply it by the number of nights entered.

Be sure to test all ranges and make sure your calculations are correct. ***Be sure to include a table of test data and expected results with your pseudocode***.

Grading Criteria:

Deliverable	Points	Breakdown
Guided exercise pseudocode	**5**	Complete, includes pseudocode and test data
Guided exercise code and run	**5**	Code is clean, runs, produces correct output
Exercise 2 (BMI) pseudocode	**5**	Complete, logic (input, process, output) is clear
Exercise 2 test data	**5**	Contains at least 4 test cases and expected results
Exercise 2 code	**5**	Code is clear, appropriate variable names, comments.
Exercise 2 run	**5**	Runs, produces correct results
Exercise 3 (Coyote Inn) pseudocode	**5**	Complete, logic (input, process, output) is clear
Exercise 3 test data	**5**	Contains at least 4 test cases and expected results
Exercise 3 code	**5**	Code is clear, appropriate variable names, comments.
Exercise 3 run	**5**	Runs, produces correct results
Lab Total	**50**	