

# CSc 110 Assignment 1:

## Introduction to Programming

### *Learning Outcomes:*

When you have completed this assignment, you should understand:

- How to design, format, document, and run a simple and complete Python program.
- The effects of escape sequences on printed strings.
- How to write and call basic functions.
- How to create and assign values to variables
- The order of control flow (i.e. the effects of function calls and assignment statements).

### *How to hand it in:*

Submit your `assignment1.py` file through the Assignment 1 link on the CSC110 conneX page.

### *Grading:*

- Late submissions will be given a **zero grade**.
- You must use the file `assignment1.py` provided to write your solution. Changing the filename or any of the code given in the file will result in a **zero grade**.
- Your function names must match exactly as specified in this document or you will be given a **zero grade**.
- None of the functions you write in this assignment should take parameters – we have not learned this yet. You will be given a **zero grade** for any functions that take parameters.
- We will do **spot-check grading** in this course. That is, all assignments are graded BUT only a subset of your code might be graded. You will not know which portions of the code will be graded, so all of your code must be complete and adhere to specifications to receive marks.
- Your code must run without errors on the ECS 258 Lab machines or a **zero grade** will be given.
- It is recommended that you use a plain text editor such as Notepad++ as used in the labs or Atom for Mac computers. We also recommend you run your programs through terminal / command prompt, as shown in the labs.
- It is the responsibility of the student to submit any and all correct files. Only submitted files will be marked. Submitting an incorrect file is not grounds for a regrade.
- If the assignment requires the submission of multiple files, then all files must be submitted. Failure to submit all required files may result in a **zero grade**.

### **Marks will be given for:**

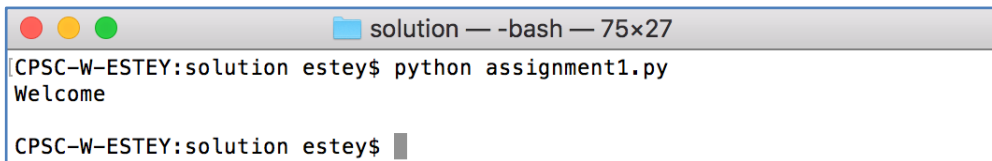
- your code producing the correct output
- your code following good coding conventions (see lab and lecture code for examples)
  - Proper indentation
  - A short **Purpose** comment above each function (we will add more steps to our function design in the upcoming weeks as our functions get more and more complex)
  - Names of variables should have meaning relevant to what they are storing
  - Use of whitespace to improve readability
  - Proper use of variables to store intermediate computation results

## Part 1 – Ascii Art

Write a set of functions that will output an ascii logo. Your implementation must include the three following functions: `print_cat`, `print_toad`, `print_spacer` and `print_logo`. You are free to introduce other functions if you would like but these four must be present and must be named **EXACTLY** as stated here or you may receive a **zero grade**.

### Step 1 – getting started:

- Download `assignment1.py` from **conneX Files** Tab and save it to your working directory (you may need to right click and select “Save Link As...”)
- Open `assignment1.py` in your editor
- Open up a command prompt (terminal window) and navigate to your working directory
- Run the program by typing at the command prompt: `python assignment1.py`
- The program will print “Welcome”. Your output should look similar to this:

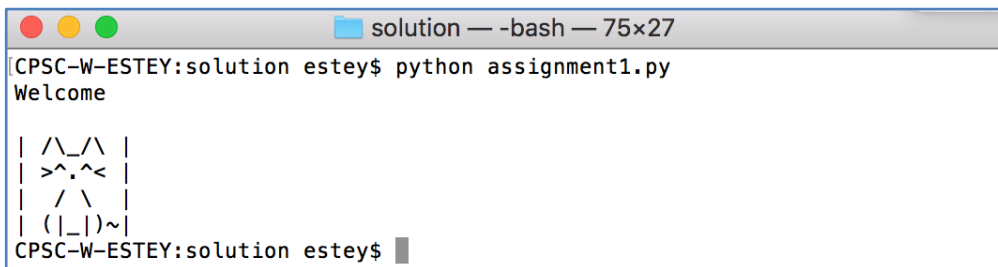


```
CPSC-W-ESTEY:solution estey$ python assignment1.py
Welcome
CPSC-W-ESTEY:solution estey$
```

### Step 2 – Write and test the `print_cat` function:

- a) In the main function of `assignment1.py`, write a call to the `print_cat` function following the existing print statements.
- b) Write the documentation and code for your `print_cat` function below the main function.
- c) Run your program from the command prompt: `python assignment1.py`
- d) Repeat steps b and c until your output looks as shown in the image below
- e) Remove or comment out the call to this `print_cat` function in your main function

NOTE: these ascii art figures looks best when your terminal uses a Courier font (ie. monospaced). Each character is composed of characters that include the: period, equal sign, double quote, forwardslash, backslash, ^, (, ), underscore and more!



```
CPSC-W-ESTEY:solution estey$ python assignment1.py
Welcome
 | /\_/\ |
 | >^.^< |
 | / \   |
 | (|_|)~|
CPSC-W-ESTEY:solution estey$
```

### Step 3 – Write and test the `print_toad` function

- Following the process from Step 2, test and write your `print_toad` function so your output looks as shown in the image below.
- Remove or comment out the call to this `print_toad` function in your main function

```
CPSC-W-ESTEY:solution estey$ python assignment1.py
Welcome

 |  @ @  |
 |  (---) |
 | ( > < ) |
 | "" "" |
CPSC-W-ESTEY:solution estey$
```

### Step 4 – Write and test the `print_logo` function

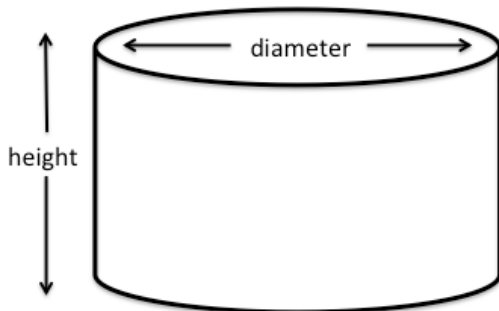
- Following the process from Step 2, test and write your `print_logo` function so your output looks as shown in the image below.
- Your `print_logo` must call the previous two functions you wrote!
- Don't forget the spacer lines that are printed between each ascii animal, which should be implemented in a separate function called `print_spacer`.

```
CPSC-W-ESTEY:solution estey$ python3 assignment1.py
Welcome

/-----\
|  @ @  |
|  (---) |
| ( > < ) |
| "" "" |
/-----\
/ \_/_\
|>^.^<|
| / \ |
| (|_|)~|
/-----\
|  @ @  |
|  (---) |
| ( > < ) |
| "" "" |
/-----\
/ \_/_\
|>^.^<|
| / \ |
| (|_|)~|
/-----\
/-----\
CPSC-W-ESTEY:solution estey$
```

## Part II – Math Calculations

Write a function `calc_surface_area` that will calculate the surface area of a cylinder given the height of the cylinder is 6m and the diameter is 5m (diagram shown below).



Ensure you adhere to the function specifications described here:

- In the main function of `assignment1.py`, write a call to `calc_surface_area` below the call to `print_logo`
- Write the documentation and code for the `calc_surface_area` function as follows:
  - Create 2 variables: one for height and set it to 6 and another for diameter and set it to 5.
  - Using the diameter and the value of PI, write a statement to calculate the circumference of the cylinder and store the result to a variable that you create.
    - For the value of PI you are free to: create a constant for PI or use the PI constant in Python's math library if you are familiar with this but it is not necessary.
  - Using the diameter and the value of PI, write a statement to calculate the area of the top of the cylinder and store the result to a variable.
  - Using the height and the circumference you calculated earlier, write a statement to calculate the area of the walls of the cylinder.
  - Using the 2 variables created above that are storing the area of the top and the area of the walls, write the lines of code to calculate the total surface area of the cylinder. Print the result to the console.
  - Don't forget – the cylinder has a top and a bottom 😊

**NOTE:** There are other algorithms to calculate the surface area of a cylinder but we are asking you to follow this algorithm to give you practice with the creation and use of variables.

Edit and run your program until your output is as shown in the image on the following page.

The accuracy of your calculation can vary as it will be dependent on the value you used for PI. In the test run shown we defined PI with 2 significant figures (3.14)

```
solution — -bash — 75x27
CPSC-W-ESTEY:solution estey$ python assignment1.py
Welcome

/-----\
|  @  @  |
|  (---)  |
| ( > < ) |
| "" ""  |
/-----\
| /\_/\  |
| >^.^<  |
| / \    |
| ( |_|)~ |
/-----\
|  @  @  |
|  (---)  |
| ( > < ) |
| "" ""  |
/-----\
| /\_/\  |
| >^.^<  |
| / \    |
| ( |_|)~ |
/-----\
133.45
CPSC-W-ESTEY:solution estey$
```