Homework 1

Intro to Programming (Term III/2024–25)

built on 2025/04/23 at 09:41:20 **due:** *wed Apr 30 @ 11:59pm*

This assignment will introduce you to the programming environment you will be using for this course and to programming in Python, as well as to our general infrastructure. In this assignment, you will install Python and PyCharm (or your Python IDE of choice), write a few simple Python programs, solve a number of programming puzzles, and hand them in.

Be sure to read this problem set thoroughly, especially the sections related to collaboration and the hand-in procedure.

Overview:

Problem	File Name
1.	–
2.	`hello.py`
3.	`types.txt`

Problem	File Name
4.	`decipher.txt`
5.	`area.py`
6.	`secret.py`

Collaboration

We interpret collaboration very liberally. You may work with other students. However, each student ***must*** write up and hand in his or her assignment separately. Let us repeat: You need to write your own code. You must not look at or copy someone else's code. You need to write up answers to written problems individually. The fact that you can recreate the solution from memory will be taken as proof that you actually understood it, and you may actually be interviewed about your answers.

Be sure to indicate who you have worked with (refer to the hand-in instructions).

Logistics

We're using a script to grade your submission before any human being looks at it. Sadly, the script is not as forgiving as we are. *So, make sure you follow the instructions strictly.* It's a bad omen when the course staff has to manually recover your file because the script doesn't like it. Hence:

- Save your work in a file as described in the task description. This will be different for each task. **Do not save your file(s) with names other than specified.**
- Before handing anything in, you should thoroughly test everything you write.
- You will upload each file to our submission site `https://assn.cs.muzoo.io/` before the due date. Please use your SKY credentials to log into the submission site. Note that you can submit multiple times but only the latest version will be graded.
- For some task, you will be able to verify your submission online. Please do so as it checks if your solution is gradable or not. Passing verification does not mean that your solution is correct, but, at least, it passes our preliminary check.
- At the beginning of each of your solution files, write down the number of hours (roughly) you spent on that particular task, and the names of the people you collaborated with as comments. As an example, each of your files should look like this:

```
# Assignment XX, Task YY
# Name: Eye Loveprograming
# Collaborators: John Nonexistent
# Time Spent: 4:00 hrs

... your real program continues here ...
```

- The course staff is here to help. We'll steer you toward solutions. Catch us in real-life or online on Canvas discussion.

Task 1: Install Python and PyCharm on Your Computer (10 points)

We will be using an IDE called PyCharm for this class. Please follow the steps from `https://python.cs.muzoo.io/protected/online-join/` and learn more about Python in PyCharm from `https://www.edureka.co/blog/pycharm-tutorial` If you are using another IDE, you will need to figure out how to run Python programs in your IDE on your own. There is no need to submit anything for this problem.

Task 2: Write a Simple Python Program (10 points)

For this task, save your work in `hello.py`

The goal of this programming task is simply to get you more comfortable with using an IDE and to begin using simple elements of Python.

You will write a program that does the following, in order:

1. Display the following prompt message: `Please enter your name:` and ask for the user's name.
2. Store the user input in a variable `name`.
3. Print out "Nice to meet you," followed by the above name (refer to the variable; don't retype the name), then by as many exclamation marks as the number of letters in the entered name. For example, if the name is "Me-ow", then you should print 5 exclamation marks.

The expected behavior of this program should be the following:

```
Please enter your name: Max
Nice to meet you, Max!!!
```

Please stick to this script to facilitate automatic grading. Do not change any wording or any punctuation. Pay attention to number of spaces and case sensitivity.

Hints

To complete this task, you will need to learn to instruct the computer (1) to print out the values you specify and (2) to read input from the console, (3) get the number of characters in a string, and (4) repeat a set of character without using looping. Python 3 has predefined functions for these; they are called **print** and **input**, **len**, and a built-in operator, respectively. You will also want to store your input for further processing. For this, you'll keep it in a variable. You can review these concepts and learn more about them from the following links:

- **print** — `https://cscircles.cemc.uwaterloo.ca/`
- **input** — `https://cscircles.cemc.uwaterloo.ca/5-input/`
- Variables — `https://cscircles.cemc.uwaterloo.ca/1-variables/`
- **len** — `https://developers.google.com/edu/python/strings` (only the "String Method" section)
- String operators — `https://www.pythoncentral.io/use-python-multiply-strings/`

Task 3: Types and Values (10 points)

For this task, save your work in `types.txt`

Part I

Assume that we first execute the following statements

```
width = 30
height = 12.0
```

Therefore, at the start, we have two variables `width` and `height` set per above.

For each of the following expressions, write the value of the expression and the type (of the value of the expression). The point of this exercise is to give you practice in predicting the type and the outcome of a Python expression, without actually running it.

1. `width/3`
2. `width/2.0`
3. `height//3`
4. `height//3.0`
5. `height*width/2`
6. `width/2*height`
7. `3 + 4 * 5`
8. `3//4 + 4*5`
9. `3/4 + 4 * 5`
10. `3/4.0 + 4 * 5`

Part II

For this part, you may use Python to help you. Here is the code that you need to trace changes.

```
1 c = 8
2 e = 6
3 d = e ** 2
4 a = d + e
5 C = a // 2
6 a = C + a
7 b = (a + c + d + e) / 4
8 e = a + b * C / d % e
9 c = c + (a - b) * e
```

From the code above, answer the following questions.

1. What is the type and the value of `a` right after line 4?
2. What is the type and the value of `c` right after line 5?
3. What is the type and the value of `a` right after line 6?
4. What is the type and the value of `e` right after line 8?
5. What is the type and the value of `c` right after line 9?

(**Remarks:** Variables in Python are case-sensitive. The variables `c` and `C` above are different variables and not a typo.)

Task 4: Decipher What's Going On (10 points)

For this task, save your work in `decipher.txt`

This problem will give you practice reasoning about Python programs. To this end, resist the urge to find answers by trying out all possibilities in Python. Instead, it is important to exercise logical reasoning to arrive at your answers (before checking them by running the code). There are two puzzles.

Program I: Find *all* settings of `x` and `y` of type **bool** that will cause this code to print `True`.

```
x = ... # bool
y = ... # bool
print(not x or y)
```

Program II: Find the *smallest* integer `x` such that the following code will print `True`.

```
x = ...
print(x >= 42 and (x%20)//10 == 1)
```

Task 5: Stadium Area (10 points)

For this task, save your work in `area.py`

A football stadium has the shape as shown in Figure 1:

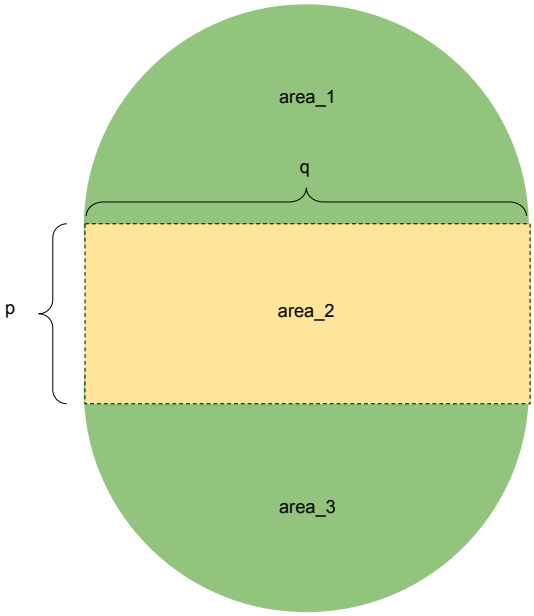

Figure 1: Football stadium illustration

The total area of the stadium comprises of 3 parts indicating by `area_1`, `area_2` and `area_3` in the figure. `area_1` and `area_3` are half-circles whose diameters are `q`. `area_2` is a rectangle of size p by q.

Your task is to write a program that prints out the total area of a stadium given the value of p and q. Your program starts by reading in values of p and q on a separate lines from the console. Here is an example input.

```
10.2
15.0
```

This means p is 10.2 and q is 15.0. The expected output should be formatted as follows:

```
The total area is 329.625
```

When we grade your program, we will change the values of p and q so it is important that your program should work for any values of p and q.

Note: You may assume that the value of $\pi = 3.14$

Task 6: Secret Code (10 points)

For this task, save your work in `secret.py`

According to the oracle of big snakes, the following number M is magical

$$M = (25 \times 1502)^{(257+123)} + (98 \times 34)^{(981-813)}$$

for some unknown reasons. Let's look at a few digits at the beginning and at the end of M:

$$M = 22479037368200 \cdots \text{middle part omitted} \cdots 42714494976$$

To tap into its magic, you will follow a simple procedure (use Python to help you):

(1) First, you will need a secret key which is an integer given to you as a user input. We'll call this k.

(2) Then, you will form your secret code by locating two digits of M. To find the first digit, which we'll call a, you will examine the digits of M from left to right. Set a equal to the k-th digit of M that you encounter. The first (i.e., left-most) digit here is 2, the second digit is 2, and so on.
To determine the second digit, which we'll call b, you will examine the digits of M from right to left. Set b equal to the k-th digit of M that you encounter. The first (i.e., right-most) digit here is 6, the second digit is 7, and so on. Finally, create your secret magic code by writing down b—followed by a.
Example: Continuing with $k = 4$ from above, we find that $a = 7$ (no, it's not 9) and that $b = 4$ (no, it's not 9). Hence, your secret magic code is 47.

Your Task: Write a program that reads a secret key from the console and prints out the secret code. Here is an example input.

```
Secret key: 4
```

Note that the prompt `Secret key:` is optional. The user only needs to type 4. The expected output should be formatted as follows:

```
The secret code is 47
```

When the secret code is a single digit or 0, you should print it out without a leading zero. For example,

```
Secret key: 14
The secret code is 0
```