HW 1 - Python Syntax + Command Line

Due February 4th, 2026 at 11:59pm

Overview

In this homework assignment, you will write a Python script on the terminal, and practice running it on both the terminal and VS Code.

Note: When you submit your homework, you will need to turn in **the files with your code and screenshots**. More information on this below.

Note: Programming is all about attention to detail! Please read the following instructions carefully.

Note: If you need help with this homework assignment, please reference **Section 5: Bonus Content** in the [**Class 2**] slides on bCourses. These bonus slides will guide you through a few problems below.

1 Create a Folder

Windows Users: Please make sure you are using Git Bash and NOT PowerShell.

In class, you created a folder structure that looks like:

```
python_decal_fa25/yourname/
```

In the `yourname/` directory, please make a new folder called `homework1/`. Type the following on the command line:

```
mkdir homework1
```

This is the folder where you will store the Python script you will work on throughout this assignment and the screenshots you take along the way. Move into the new `homework1/` directory and the call `pwd` to show your entire file path.

Take a screenshot of you making this directory on the command line. Name it `hw1_folder`. Store it in your `homework1/` directory.

2 Make a New File in VS Code

1. Open VS Code.
2. Navigate to your `homework1/` folder.
3. Create a new file called `homework1.py`.

3 Write a Python Script

At the top of your file, add the following line:

```
# File: homework1. py
```

It's good coding practice to add a comment line (notated by the #) at the top of your file stating its name. It will help keep you more organized.

Section Headers

At the top of each of the follow sections, please add a header like the following:

```
# --- Variables and Data Types ---
```

3.1 Variables and Data Types

In class we covered three data types: strings, integers and floats. Now you will work through several more. For **each** variable given below:

1. Print the variable.
2. Print the data type of the variable.
3. Leave a comment describing what kind of data it is.
   Hint: Use the internet.

Example:

```
a = 10
print(a)
print(type(a)) # a is an integer, a whole number with no decimals
```

Variables:

- `a = 10`
- `b = 1.5`
- `c = 3j`
- `d = "hello"`
- `e = [1, 2, 3]`
- `f = {"name":  "Ellen", "favorite fruit":  "strawberry"}`
- `g = (1, 2)`
- `h = ["apple", "banana", "strawberry"]`

- `i = True`
- `j = None`
- `k = [True, "blue", 12]`
- `l = str(14)`
- `m = 1e4`

Questions:

Answer the following questions as comments below your variables.

1. How many different data types did you find?
2. List all the data types you found.
3. What variables have the same data types?
4. What was the data type of `l`? Why is it not an integer? What does `str()` do?
5. Look up **one more** data type not given above. Repeat the same procedure.

3.2 Booleans

A **boolean** is a data type in Python which returns `True` or `False` depending on your code. Run each line given below. Next to the line, write a comment if it was `True` or `False`. Describe the result.

Example:

```
print(10 > 9) # True, 10 is greater than 9
```

Expressions:

- `10 > 9`
- `10 == 9`
- `10 <= 9`
- `bool("abc")`
- `bool(123)`
- `bool([‘‘apple", ‘‘cherry", ‘‘banana"])`
- `bool(True)`
- `bool(False)`
- `bool(0)`
- `bool("")`
- `bool(" ")`
- `bool(())`
- `bool([])`
- `bool({})`

- `bool(True and False)`
- `bool(True and True)`
- `bool(False and False)`
- `bool(True or False)`
- `bool(True or True)`
- `bool(False or False)`
- `bool(not(False))`
- `bool(not(True))`

Questions:

- What pattern do you notice about expressions returning `True` or `False`?
- Which expression surprised you about its result?
- Create an expression, not given above, that will return `True`. Why is it True?
- Create an expression, not given above, that will return `False`. Why is it False?

3.3 Operators

An **operator** is a special symbol or keyword that performs operations. We will explore the following types of operators:

- Arithmetic
- Comparison
- Assignment
- Logical

Run each line below, leave a comment next to it with the result and a description.

Example:

```
print(10 + 5) # 15, + performs addition
```

3.3.1 Arithmetic Operators

- `10 + 5`
- `10 - 5`
- `2 * 4`
- `6 / 3`
- `5 % 2`
- `3 ** 2`
- `15 // 2`

3.3.2 Comparison Operators

- `5 == 2`
- `10 != 10`
- $2 < 5$
- $12 > 5$
- $5 <= 6$
- $1 >= 10$

3.3.3 Assignments Operators

Create a variable: `x = 5`. Perform the following operations on it.

- `x += 5`
- `x -= 4`
- `x *= 3`

3.3.4 Logical Operators

Answer the following questions as comments:

1. What does the operator `and` do? Write an expression that results in `True`. Write an expression that results in `False`.
2. What does the operator `or` do? Write an expression that results in `True`. Write an expression that results in `False`.
3. What does the operator `not` do? Write an expression that results in `True`. Write an expression that results in `False`.

More Questions:

1. What is the difference between `/` and `//`?
2. What is the difference between `%` and `//`?
3. What operator would you use to calculate the remainder when dividing two numbers? Give an example.
4. How do assignment operators work?

3.4 Strings

Start with the string `my_string = "hello"`. Perform the following mutations and describe the results with a comment.

Example:

```
my_string = "hello"
print(my_string) # Prints: hello
```

Mutations:

1. Print `my_string`.
2. Print `my_string[0]`.
3. Print `my_string[1]`.
4. Print `my_string[2]`.
5. Print `my_string[3]`.
6. Print `my_string[4]`.
7. Print `my_string[-1]`.
8. Print `my_string[1:3]`.
9. Print `my_string[0:5:2]`.
10. Print the length of the string: `len(my_string)`.
11. Print the result of adding `my_string` with ”goodbye”.
12. Print the result of multiplying 7 times `my_string`.

3.4.1 Questions:

1. Define the term slicing. For which of the manipulations did you slice your string?
2. Call the following, describe the result:

   ```
   name = "Oski"
   print("Hello, my name is", name)
   ```

3. Call the following, describe the result.

   ```
   name = "Oski"
   print(f"Hello, my name is {name}")
   ```

4. What is the difference between the two last print statements?
   Hint: Look up f-strings.

3.5 Terminal Commands

For each command listed below, do the following:

1. Define the command as a comment.
2. Explain how to use it.
3. Give an example of how to use it.
4. Try out the command on your terminal.

Hint: If you don't know one, feel free to look it up!

Example:

```
# cd
# Changes directories. Use it to move from one folder to another
# Example: cd Desktop
```

Commands:

1. cd
2. ls
3. ls -a
4. mkdir
5. cat
6. pwd
7. cd ..
8. cd .
9. cd ~
10. cp
11. mv
12. rm *(be careful with this one)*
13. clear
14. grep

Questions:

1. Look up 3 other commands not present. Define and explain how to use them on the command line.
2. What is the difference between `ls` and `ls -a`?
3. What is a hidden file?
4. Look up 3 other **flags** (e.g., `-a` was a flag for the `ls` command). Define and explain how to use them on the command line.

4 Running Your Script

Please complete all the parts above before continuing.

4.1 In Your VS Code Terminal:

1. Open VS Code.
2. Run your **completed** script with the arrow button.
3. Take a screenshot of your **code** and the output on the VS Code **terminal**.
4. Name the screenshot: `hw1_vscode`
5. Save the screenshot in your `homework1/` folder.

4.2 On Your Terminal Application:

1. Open the terminal (Mac) or Git Bash (Windows).
2. Navigate to the directory that holds your `homework1.py` script.
   Hint: It should be located in `python_decal_fa25/yourname/homework1/`.
3. Call `pwd` and `ls`, to show your current working directory and to list all the files and folders inside.
4. Take a screenshot of your `pwd` and `ls` commands on the terminal.
5. Name the screenshot: `hw1_pwd_and_ls`
6. Save the screenshot in your `homework1/` folder.
7. Run your completed `homework1.py` script by calling:

   ```
   python3 homework1.py
   ```
8. Take a screenshot of your **entire** script output.
9. Name the screenshot: `hw1_commandline`
10. Save the screenshot in your `homework1/` folder.

5 Submitting Your Homework

Before submitting this assignment to Gradescope, make sure your `homework1/` folder looks like this. You should have **one** `homework1.py` file and **four** screenshots:

```
homework1/
|--- homework1.py
|--- hw1_folder.png
|--- hw1_vscode.png
|--- hw1_pwd_and_ls.png
|--- hw1_commandline.png
```

Please make sure your screenshots are named as shown. This will make it easier for the graders.

Upload your **entire** `homework1/` folder to Gradescope. Great job!