I/O & Variables Project

A supplementary Python project for ECS by Jon Stapleton & Blythe Samuels

Summary

In this project, students create a program that performs calculations on input values to produce formatted output. Students will choose a project to create from a list of three options, or generate their own option that meets the activity requirements.

Objectives

The students will…

- Design, test, and revise Python programs that…
  - Perform arithmetic operations variables
  - Take user input and store them in variables
  - Generate formatted output based on user input
- Evaluate the role of data collection in contemporary computing
- Evaluate the kinds of data & decision-making easily represented by computing systems, and those that are excluded by the limitations of computing hardware, data types, etc.

Standards Alignment

- **CSF.12** The student will develop a program working individually and in teams using a text-based language.
- **CSF.13** The student will identify the expected output of a program given a problem and some input.
- **CSF.15a** The student will design and implement algorithms using sequencing of instructions.
- **CSF.16** The student will implement a program that accepts input values, stores them in appropriately named variables, and produces output.
- **CSF.18** The student will apply the basic operations used with numeric and non-numeric data types in developing programs.

Materials

- Python IDE (e.g., Repl.it, Thonny, Mu, or a text editor and the command line)
- The *1.3 Input, Output, & Variables Project Handbook* (PDF, view Google Slides, or make a copy)
- *Input & Output Practice Project Description & Rubric* (see p. 5-7 in the *Handbook* or below)
- Input & Output project exemplars: Change Exchange (.py), Gas Money (.py), Group Order (.py)

Outline

Formative Assessment Notes

1. **Primer Activity:** Run an example program that makes use of input, output, variable(s), and arithmetic operations (see *Project Prompts* below for code to use, or make your own). Then, lead a discussion around the following questions:

   - ***What input does the user provide to the program?***
   - ***What output does the program generate?***
   - ***What sort of task does the program accomplish?***
   - ***What calculations take place in the program to produce the program's output?***

   Consider having students discuss in pairs and in fours before discussing with the class as a whole.

   Use this as a moment to check with students to make sure everyone has a solid understanding of input & output.

   Consider having students complete a self assessment (see below or previous lesson) to inform the rest of the lesson if you haven't already had them complete one.

2. **Program Analysis:** Have students work in small groups (2-4) to write pseudocode (e.g., plain language, flowchart) for the example program, trying to figure out what commands the program might contain in its source code.

   Emphasize expressing the process the computer goes through rather than recalling syntax rules. Don't spend too much time here—make sure your example isn't too complicated.

   If students are stuck consider prompting them:

   - **How many print()s do you think this program contains?**
   - **How many input()s?**
   - **How many variables?**

3. After students write their pseudocode, show them the example source code and discuss. You can facilitate this discussion with the whole class, or prompt small groups to discuss:

   What are the differences between your pseudocode and the source code? Does anything in the source code surprise you?

   Use this moment to assess students' understanding of input, output, and variables. Re-teach concepts using the code example as necessary.

4. **Project Introduction:** Introduce the *Input & Output Project* (see *Assessment Strategies* below), answer student questions, and provide an overview of how students will spend their time while working on the project.

   See *Assessment Strategies*

5. **Project Work:** Have students spend several days working on their projects. At the end, have them share their work (see *Assessment Strategies* below).

   See *Assessment Strategies*

Assessment Strategies

In addition to formative assessments (see *Outline* above), here are a few summative assessments:

Self-Assessment — *See below or p. X of the 1.2 Handbook for printable versions*

Name: _____ Date: _____

Rate yourself on the following skills:

I can write code that produces text output	0 1 2 3 4
I can write code that stores input in a variable	0 1 2 3 4
I can write equations that include variables	0 1 2 3 4
I can produce formatted output that includes variable values in the outputted text	0 1 2 3 4

Journal Entry

College admissions departments often use computer programs to help them decide which students they will admit to their freshman class. Imagine a program that asks students for the following information, and uses the numbers students provide to calculate a score:

- Grade point average
- Attendance percentage
- Number of extracurricular activities

In the above hypothetical situation, only the top 10,000 students with the best scores will be allowed to submit a full application, which includes an essay and personal statement that an admissions committee will review. Given what you know about this application process, address the following questions:

1. How many variables would this program need to have? What type (string or number) are they?
2. Why is it difficult for a computer to analyze an essay the same way a human reviewer would? Why is it easier for a computer to calculate a score for applications compared to a human reviewer?
3. Is this automated system just? Why or why not?

4. This system may exclude students who would otherwise be admitted based on their unique circumstances. How would you improve this system to make it more fair, making sure everyone has an equal chance to go to college and enjoy the opportunities it offers?

Input, Output & Variables Project

In this project, students will create a Python program that meets the following criteria:

- Has at least 3 variables, to hold values collected via the input() function
- Performs an arithmetic calculation on the input values
- Displays the results of the calculation in an appropriate format using print()

Consider offering the project to students in one of the following ways:

- ☐ **Create Your Own:** Create a program that expresses something about you, something about your interests, or that meets a personal need. Collaborate with your teacher to design your project so it meets the assignment parameters using the *Project Design Menu* (see see note below)
- ☐ **Remix an Example:** Choose one of the example project descriptions below, and "remix" it so it fits your interests, goals, or needs.
- ☐ **Choose an Option:** Choose one of the example project descriptions below, and create a program that matches it.

Options	Description
Change Exchange	Create a program that will calculate the total value of an amount of coins. The program calculates the total dollar amount for the number of quarters, dimes, nickels, and pennies entered by the user (see example)
Gas Money	Your friends are bothering you for a ride to school. Create a program that will calculate how much money they owe you for gas. The user should input miles to school and the MPG for the car (see example)
Group Order	Create a program that calculates the total price of a meal for a group. The program should prompt the user to enter how many of each type of item they want: drinks, appetizers, main meals, and desserts (see example)

Students can use the *Project Guidelines* (see below) as a reference for what they should produce.

Using the Project Design Menu: The project design menu is a flexible project planning tool to scaffold students' project design choices. To use the tool, assign an arbitrary "target" value (the menu provided below uses a target of 30 points) and ask students to select goals from the list on the menu to meet or exceed the target value. We have provided a list of "easy" 3-point goals, and a list of more "difficult" 5-point goals. The menu we created for this project also asks students to create their own goal based on an open-ended prompt for 12 points. The blank lines are there so you (or ideally, students) can add their own "easy" and "difficult" goals and claim them. You should encourage students to create goals that help them get credit for creating programs that are useful, interesting, or fun. See p. 6 in the *Handbook*.

Note: Some of the "difficult" goals require learning new concepts, like randomization or using time.sleep()

Gallery Walk

After students have created their programs, consider facilitating a gallery walk. Ask students to answer the following questions about one or more projects during the gallery walk (see p. 8 of the *Handbook*):

- What input does the program provide? What output does it produce?
- How many variables do you think the program includes? Explain your reasoning.
- Do you think the program includes any expressions? If so, what do you think they calculate?

Input, Output & Variables Project Rubric

Consider using the rubric below to communicate the project requirements to students, and to assess student work at the end of the project:

	Proficiency	*Yes*	*No*	*Notes*
Input	Student's program accepts input and stores it in a variable			
Output	Student's program produces output, including formatted output			
Variables & Types	Student's program includes variables, and casts input values as appropriate data types when necessary			
Expressions	Student's program makes use of expressions, including expressions that use variable values as operands			

Some Accommodations & Extensions

Note: All students benefit from accommodations; consider implementing the accommodations below for everyone

Accommodations

Some students may benefit from starting their project with a few lines of code. Modify the example programs (see *Materials*) if they are trying to replicate the example projects, or give students the examples as a starting point if they are creating an original project.

Consider having students read and write pseudocode for their programs before starting the Python code.

Extensions

Use the *Project Design Menu* in the *Handbook* to prompt advanced students to set ambitious goals for themselves. You can do the following to set this up:

- Increase the "target" number of points they need to claim from 30 to a higher number to get them to design a more ambitious project
- Add additional feature requests to the example projects
- Require students to choose the randomization and time.sleep() prompts from the menu and implement these new libraries. Be sure to provide additional instruction or the library documentation
- Have advanced students create more than one project with no overlapping menu items

This unit has a low ceiling, so don't worry if some students complete the work very quickly—there will be lots of complexity to take on board in future units.

Input, Output & Variables Project Guidelines

In this project, you will create a Python program that meets the following criteria:

- Has at least 3 variables, to hold values collected via the input() function
- Performs an arithmetic calculation on the input values
- Displays the results of the calculation in an appropriate format using print()

Here are some examples of projects that will meet the requirements above:

Options	Description
1 Change Exchange	Create a program that will calculate the total value of an amount of coins. The program calculates the total dollar amount for the number of quarters, dimes, nickels, and pennies entered by the user.
2 Gas Money	Your friends are bothering you for a ride to school. Create a program that will calculate how much money they owe you for gas money. The user should input the distance to school for each friend and the MPG for the car
3 Group Order	Create a program that calculates the total price of a meal for a group. The program should prompt the user to enter how many of each type of item they want: drinks, appetizers, main meals, and desserts.

Project Options

To complete the project, choose one of the options below:

- ☐ **Create Your Own:** Create a program that expresses something about you, something about your interests, or that meets a personal need. Collaborate with your teacher to design your project so it meets the assignment parameters using the *Project Design Menu* (see back of this page)
- ☐ **Remix an Example:** Choose one of the example project descriptions above, and "remix" it so it fits your interests, goals, or needs.
- ☐ **Choose an Option:** Choose one of the example project descriptions above, and create a program that matches it.

Input, Output & Variables Project Rubric

	Proficiency	***Yes***	***No***	***Notes***
Input	Student's program accepts input and stores it in a variable			
Output	Student's program produces output, including formatted output			
Variables & Types	Student's program includes variables, and casts input values as appropriate data types when necessary			
Expressions	Student's program makes use of expressions, including expressions that use variable values as operands			

Printable Self-Assessment Tickets

Name: _____ Date: _____

Rate yourself on the following skills:

I can write code that produces text output	0 1 2 3 4
I can write code that stores input in a variable	0 1 2 3 4
I can write equations that include variables	0 1 2 3 4
I can produce formatted output that includes variable values in the outputted text	0 1 2 3 4

Name: _____ Date: _____

Rate yourself on the following skills:

I can write code that produces text output	0 1 2 3 4
I can write code that stores input in a variable	0 1 2 3 4
I can write equations that include variables	0 1 2 3 4
I can produce formatted output that includes variable values in the outputted text	0 1 2 3 4