

BASICS: *Biophysics - A Step-by-step Introduction to Concepts for Students*

Lesson Plan: Writing Computer Code

Background

Computers are everywhere! Besides the personal computer on your or your parent's desktop, computers are in your smart phone, PlayStation for video games, MP3 player and television remote control. Other examples of computers that you encounter everyday are computer systems in cars that ensure engines start and run smoothly, as well as those that provide directions via a Global Positioning System, or GPS. How do these computers know what they are supposed to do? The answer is simple: someone has **programmed** the instrument to do its task – this means that a person who knows **computer code** has written instructions and placed them in the instrument to perform its task.

Different **computer programming languages** can be used to set tasks for machines. There are many programming languages, including C, Python, C++, and JavaScript. These languages are similar to those that you and I speak, for example, English, Spanish, Mandarin, Hindi, and others. However, programming languages differ from spoken languages in that they allow you to communicate with a computer. You need to understand simple rules to write code in a given programming language. After learning how to write code, you will be able solve problems, program the thermostat in your parents' home, and design your own video games!

Writing computer code is a valuable part of being a biophysicist, as it can make analysis of experimental data much easier and faster. This is important for large datasets consisting of numbers or images, but it can also be used to save time in analyzing smaller datasets or solving problems involving mathematical calculations.

Objectives & Grade Level

Teach students basic features of the Python computer programming language, enabling them to write and run short **scripts**, or lists of computer instructions. Appropriate for middle school to high school science classes; see **Notes** for advanced students.

Materials

- Personal computer, e.g., laptop, or iPhone or another smart phone
If you have a personal computer, you can use a web browser (e.g., Safari, Firefox, Chrome, Microsoft Edge, etc.) to open the Online Python link below. You can also use an iPhone or another smart phone to open Online Python and perform the experiments in this lesson plan.
- Online Python at <https://www.online-python.com/>
*Scripts are written in an **Integrative Development Environment**, or **IDE**. There are many Python IDEs available that you can download onto a computer to use, e.g., PyCharm, Spyder, and IDLE. However, instead of downloading an IDE, you can open an online IDE from a web browser – this works well when you are learning the basics of Python. The one we will use in this lesson plan is called Online Python.*

Procedure

Experiment 1: Writing & Printing a Script

Think of coding like giving someone instructions, in this case, a computer. You can tell the computer what to do by using special commands and **functions**, words that carry out a specific task. The list of instructions is called a **script**, like scripts given to actors in plays or movies. In this experiment, you will learn how to write and print a script.

1. Type <https://www.online-python.com/> into the search line of your web browser and press Return
2. A window will appear like the one below in **Figure 1**. Each part of the window has a specific function, which we explain in the next five steps.
3. First, change the name of your file to “MovieScript.py” by double-clicking in the box where it says “main.py” This is the name of your script. You should name your script after what it does, so you can find the script when you need it later. Note that **you should not use spaces in your file name**. Instead, capitalize the first letter of each word and leave out the spaces. This format is called **CamelCase** because the capital letters look like a camel’s humps – ThisIsAnExampleOfCamelCase. The first letter can be capitalized or not, depending on your preference. The .py at the end of the name means that the script is a Python script.

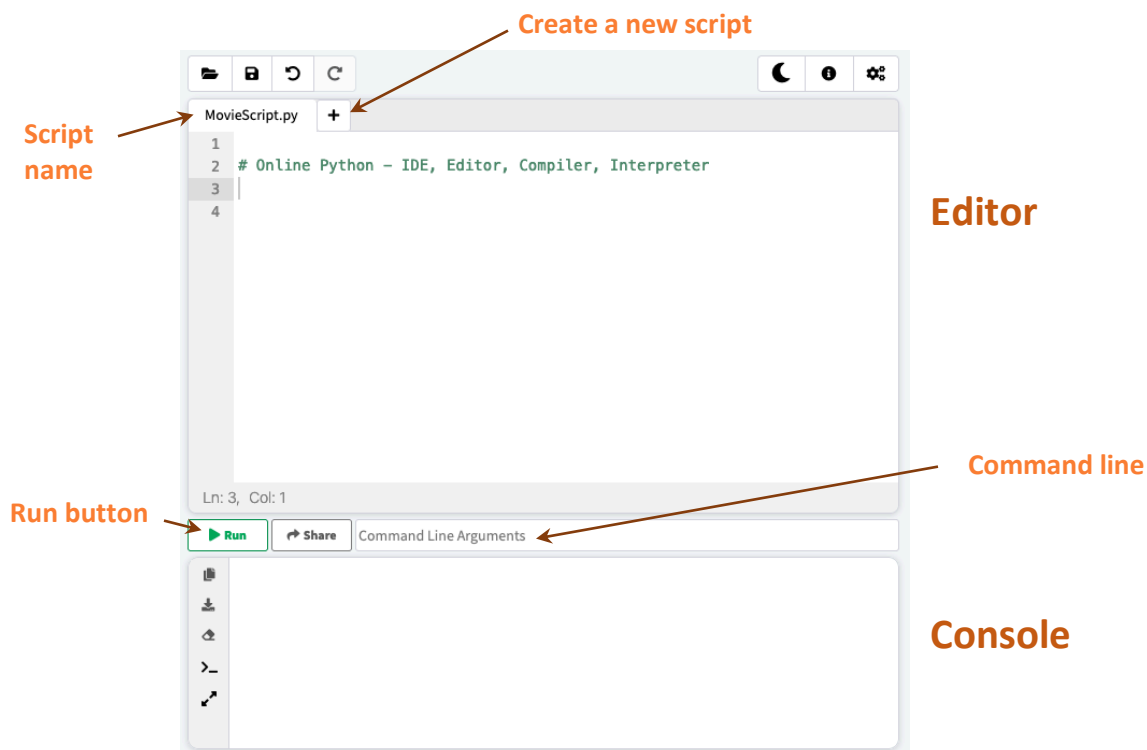


Figure 1 Layout of the Python IDE

4. Find the **Editor**, using **Figure 1** as a guide. This is the large box where you will write your script. Each line is a step in the instructions. The steps are written in order from top to bottom. There are some commands that rely on the previous step and these will be indented. You will use these as you learn more complex Python functions.
5. Find the **Run** button. This will run the commands in your script in the order that you wrote them.
6. Find the **Command line**. If you want to run a single command, you can write it in the Command line and press Enter.
7. Find the **Console**. The Console shows you the results of your script. Additionally, if there are **errors**, or mistakes in your script, an **error message** will show up in red in the Console. You can think of **coding errors** as spelling mistakes in the commands, or mistakes in the order of the steps.
8. Learn the command for **Print**. This is a simple and useful Python function. It “prints” to your Console.

To print, type

```
print(" ")
```

then type the words you want to print inside the quotation marks that are inside the parentheses. The quotation marks tell the computer what you want to print. You can use either double (“ ”) or single (‘ ’) quotation marks.

For example, if you wanted to print “Hello!”, you would type the following:

```
print("Hello!")
```

and then click on the Run button. *Hello!* will appear in the Console, as shown below in **Figure 2**:

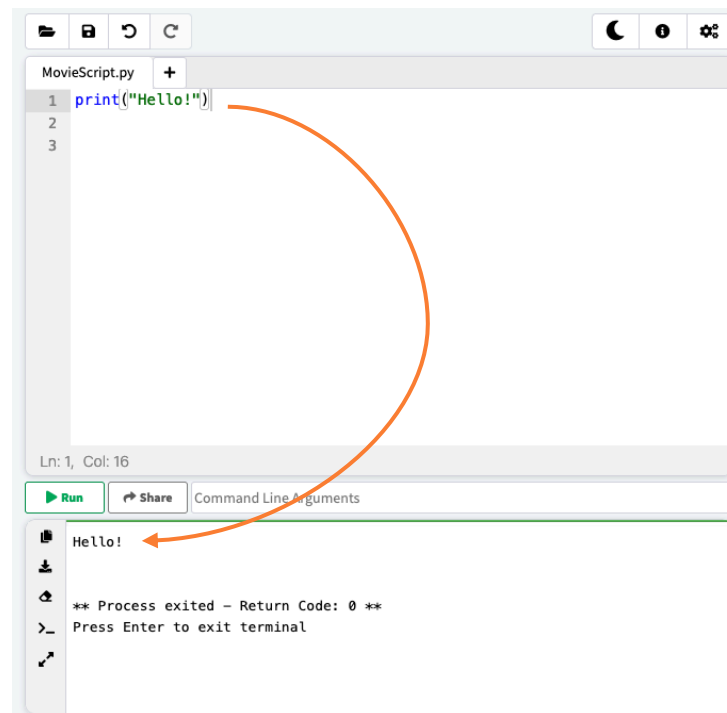


Figure 2 Print command

9. In the MovieScript.py Editor, type `print()` in Line 1 and fill in the name of your favorite movie in the parentheses. Remember to include quotation marks around the movie name. For example, if your favorite movie is Monty Python, the command would look like this:

```
print("Monty Python")
```

10. Click on Run, then look in the Console – if the name of your favorite movie is there, congratulations! You just ran your first Python script! If not, look carefully at **Figure 2** above. It shows the `print` command written correctly and the correct output in the Console.

Now try making a new script by clicking on the plus sign (+) in the Editor. Name the script “FavoriteColor.py”, then type the command to print your favorite color. Did your script print your favorite color?

Experiment 2: Learn About and Print Variables

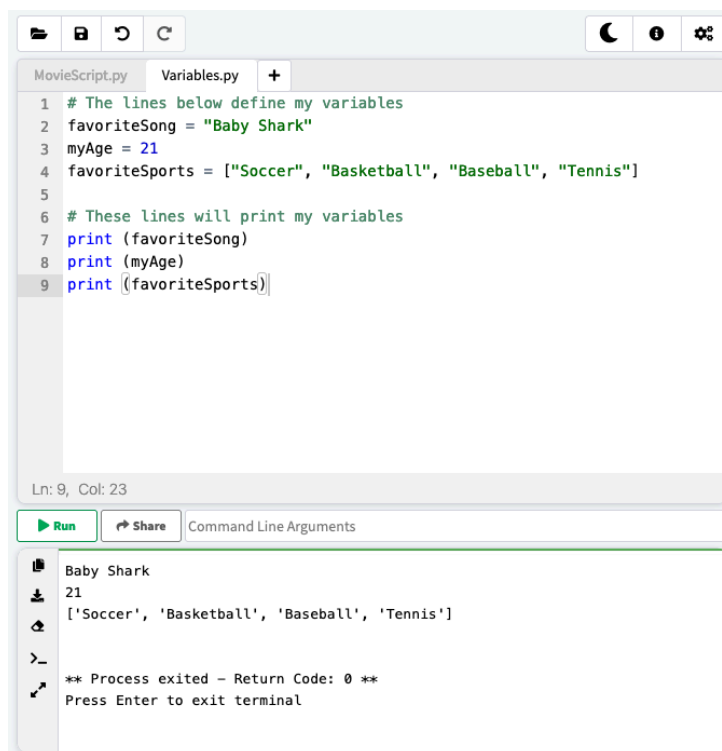
A **variable** in computer language is information that is stored under a name that you select. Three common types **variables** that are used when writing computer code are **numbers**, **strings**, and **lists**:

- **Numbers** can be integers (**int**), decimals (**float**) or complex numbers (**complex**). Examples of numbers written in different ways are 4, 4.0, 123456
- **Strings** are groups of words or characters enclosed by quotation marks. You used strings with the `print` command above. Here are some examples of strings: “Hello”, “This is a string”, “a”
- **Lists** are numbers, letters, or words separated by commas. The items in the list are enclosed by brackets; however, they do not have to be the same type and they can also include another list. Two examples of lists are [“hello”, “hola”, “bonjour”] and [101, [1,5,6], “ocean”]

Variables are defined, or named, using an equal sign. The variable names are written in CamelCase without spaces, like script names. Here are some examples (see **Figure 3**):

```
favoriteSong = "Baby Shark"  
myAge = 21  
favoriteSports = ["Soccer", "Basketball",  
                 "Baseball", "Tennis"]
```

Every time you type the variable name, Python will use the definition that is assigned to it. Variables are useful when you need to use the same information several times. Variables can be modified, changed, or redefined as needed.



```
MovieScript.py  Variables.py  +  
1 # The lines below define my variables  
2 favoriteSong = "Baby Shark"  
3 myAge = 21  
4 favoriteSports = ["Soccer", "Basketball", "Baseball", "Tennis"]  
5  
6 # These lines will print my variables  
7 print (favoriteSong)  
8 print (myAge)  
9 print (favoriteSports)
```

Ln: 9, Col: 23

Run Share Command Line Arguments

```
Baby Shark  
21  
['Soccer', 'Basketball', 'Baseball', 'Tennis']  
>  
** Process exited - Return Code: 0 **  
Press Enter to exit terminal
```

Figure 3 Variables

1. Create a new script named Variables.py.
2. Make a variable named luckyNumber for your lucky number.
3. Make a variable named favoriteColor for your favorite color.
4. Make a variable named favoriteFoods that lists your favorite foods as a string.
5. Print each variable. To print a variable, use the command `print(variableName)`. If you use more than one `print` command, list each one on a separate line.
6. Look at the console to see if the variables were printed correctly.

What happens when you type `print("luckyNumber")` in your script, instead of `print(luckyNumber)`? Why do you think this happens? What can you conclude about the `print` command?

Experiment 3: Using Python as a Calculator

Much of the advanced technology that exists today is made possible by the ability of computers to perform complicated math quickly. You can use Python to do basic math using the math symbols shown below:

Operation	Symbol
Addition	+
Subtraction	-
Multiplication	*
Division	/

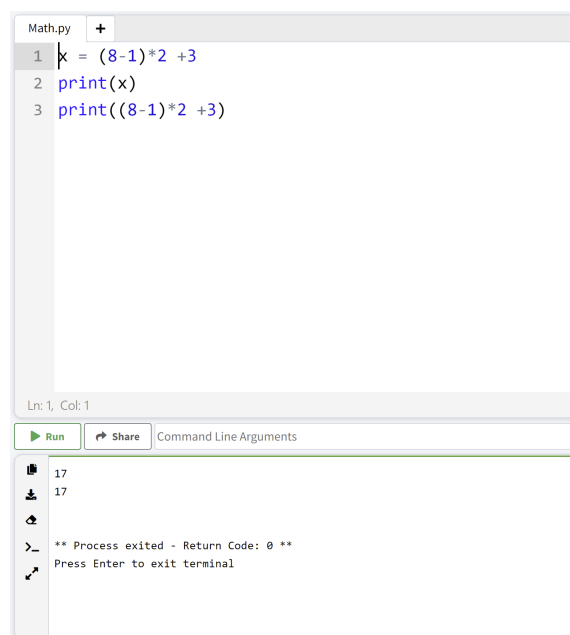
Variables can be used to define a sequence of mathematical operations. For example, if you wanted to calculate the number of candies you need to buy to give two to each of your eight friends, except your best friend, who prefers cookies, and keep three for yourself, you can calculate the number of candies as x , by typing

$$x = (8-1)*2 + 3$$

Then enter a **print command**, which is very useful for solving simple math expressions. The correct value will appear in the Console after you press Run. Note that you can enter either the command `print(x)` or `print((8-1)*2 + 3)` and the same answer appears in the Console.

1. Create a new script and name it Math.py
2. Write a `print` command for `26 + 19`. Leave out the quotation marks because you want Python to compute the answer as numbers.

What does the computer print when you enter the command `print("26 + 19")`?



```
Math.py +
1 x = (8-1)*2 + 3
2 print(x)
3 print((8-1)*2 + 3)

Ln: 1, Col: 1
Run Share Command Line Arguments

17
17

** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

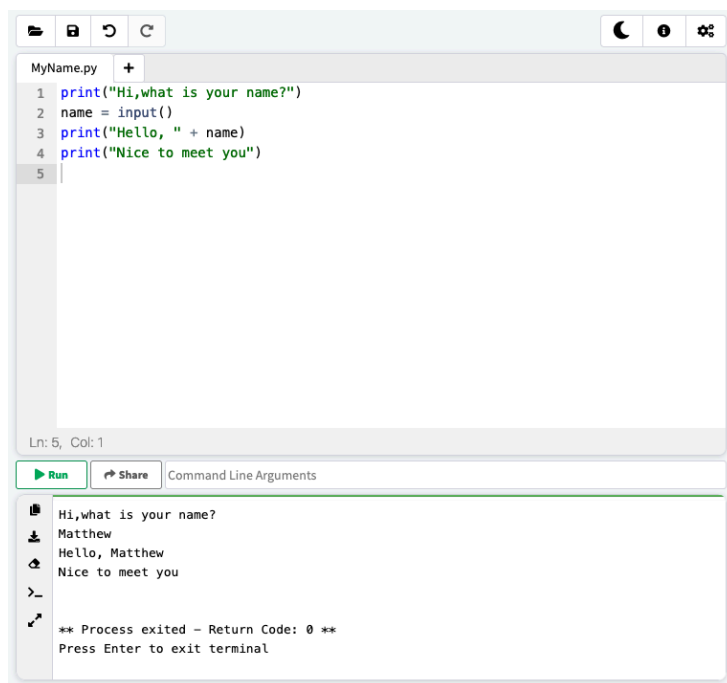
Figure 4 Math using Python

3. Press Run. Is the number in the Console what you expected?
4. Replace the numbers in the *print* command with 30/3 and press Run.
5. Replace the numbers in the *print* command with 6*8 and press Run.
6. Replace the numbers in the *print* command with 99-91 and press Run.
7. Delete all the lines in your script.
8. Define a variable named “a” equal to 25
9. Define a variable named “b” equal to 5
10. Define a variable named “c” equal to 7
11. Print a+b. This is the same as adding 25 and 5.
12. Print a-c . This is the same as subtracting 7 from 25.
13. Print b*c. This is the same as multiplying 5 by 7
14. Print a/b. This is the same as dividing 25 by 5
15. Press Run to run the script. The numbers that appear in the Console should be the values that you expect from performing the calculations.

Experiment 4: Talking to your Computer

Whenever you do a Google search or play a game, your computer or smart phone is listening and watching. What appears on your screen depends on the keys you press, as if you are talking to your computer. This experiment shows how you can write commands to make your computer or smart phone listen and talk to you.

1. Open a new script and name it “MyName.py”
2. On the first line, create a *print* command that asks for your name. For example, “Hi, what is your name?”
3. On line 2, define a new variable named *name* that equals *input()*. Line 2 of **Figure 5** shows you how to do this.
4. On line 3, type another *print* command that contains a greeting with a space after the last word, followed by the second quotation mark. Then type a plus sign and the variable name from line 2, followed by the parentheses. Possible greetings include “Hello” or “Hi”. See line 3 of **Figure 5** for a *print* command in this form. Then type another print command for “Nice to meet you”, as shown on line 4 of **Figure 5**.
5. Press Run. The first line in the Console will be the question that the computer is asking you with a blinking cursor below the question. Reply to your



```
MyName.py +
1 print("Hi,what is your name?")
2 name = input()
3 print("Hello, " + name)
4 print("Nice to meet you!")
5

Ln: 5, Col: 1
Run Share Command Line Arguments

Hi,what is your name?
Matthew
Hello, Matthew
Nice to meet you
>
** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Figure 5 Talking to your computer

computer by typing your name on the second line and then press enter. The computer will then reply to you with the greeting that you wrote into your script.

The computer knows your name because you typed your name in response to the question. What do you think input() means when writing computer code? How can you change your script to ask different questions so that the computer replies are different?

Notes

1. We omitted some computer coding terminology to make this lesson plan easier to understand for students who are just beginning to learn about Python. As you learn more about writing computer code and programming languages, the following terms will become familiar to you:
 - **Argument:** The part of a function/command that specifies what you want it to do. The argument is usually typed inside the parentheses. The general form is *command(argument)*. For example, when typing the command *print("Hello")*, "Hello" is the argument.
 - **Output:** Information produced by a command. For example, the output for *print()* is the argument, which is printed to the Console. Other outputs might be the answer to a formula, a graph, or a value that can be saved to a variable.
 - **Syntax:** The "grammar" of coding. These are the rules for how different computer codes are used. Different programming languages have different syntax, similar to the different phrase and sentence structures of different spoken languages. There are many online resources available, if you have questions about the syntax used in Python or another computer programming language.
2. **Advanced topic 1:** in **Figure 3**, the lines in which all the words appear in green are **comments**. Comments are not computer commands; instead, they are notes or statements that can be used to explain to others what your code does. Comments have a # symbol at the beginning of the line, which means that everything on that line is a comment.
3. **Advanced topic 2:** in Experiment 4, you may have noticed that we are adding a string to a string variable with a + symbol. In this case, + does not represent addition but rather **concatenation**. Concatenation combines 2 strings to form a single, longer string. This is often done when incorporating variables with strings or numbers with strings.

BASICS: *Biophysics - A Step-by-step Introduction to Concepts for Students*

Resources

Here are some web pages that will help you learn more about Python:

<https://www.w3schools.com/python/default.asp>

<https://www.learnpython.org/>

<https://docs.python.org/3/tutorial/>

<https://pythonbasics.org/>

Here are links to more advanced topics in writing computer code:

Concatenation: https://www.w3schools.com/python/gloss_python_string_concatenation.asp

Input: https://www.w3schools.com/python/ref_func_input.asp

Acknowledgements

This lesson plan was written as part of Broader Impacts of research supported by the National Science Foundation under Grant Number CMMI 1660924 to S.A.E. Any opinions, findings, conclusions or recommendations expressed in this lesson plan are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Authors

Matthew Y. Wang
Pratt School of Engineering, Class of 2023
Duke University

Sharyn A. Endow, PhD
Department of Cell Biology
Duke University Medical Center

Copyright © 2022 by Biophysical Society. All rights reserved.