# Introducing Students to Computer Science: Using Python to Build a Basic Quiz Game

Alex Diosdado

Spring Quarter, June 2022

**Abstract:**

This outreach program activity was designed to increase interest in the field of Computer Science within students in middle and high school. It focuses on some of the basic concepts of programming, like Strings, Integers and printing to the screen as it was designed to be an introduction for those students who had little to no previous programming history. This activity was tested with a smaller group of high school Algebra 1 students, most of whom did not have any prior programming knowledge. It ultimately fulfilled its goal as a survey that was conducted at the end of the activity found that some of the students who had no prior CS history felt they were more interested in taking classes in the Computer Science field.

**Introduction:**

Over time, technology has become further and further entrenched in people's daily lives. As such it is important that we understand how and why it works, no different from how we should learn about our world through topics in the sciences such as Biology, Chemistry and Physics. However, Computer Science is not seen as an important subject to high schools which leads to many students who go through their early academic careers without access to any kind of Computer education. According to CSforCA, an organization that advocates for high quality computer science education for California students, only 39% of high schools in the state of California actually offer computer science classes for their students and ⅔ of all high schools in California do not offer any classes for computer science. Ultimately, only 3% of California's 1.3 million high school students took a computer science class and only 1% of them took the APCS exam. As the tech industry continues to expand, there is a continued need for programmers and computer scientists to fill positions, but it is becoming clearer that the next generation of students are not being given the proper opportunity to learn the skills necessary for these kinds of careers before they start their university classes. Despite California being the largest economic provider out of all 50 states in the US and multiple major tech companies calling the state home, it lags far behind other states in terms of high school student participation. According to The California Computer Science Access Report, California lags behind the national average in the percentage of high schools that offer at least one CS course (~47%) and lag behind 34 other states. California's 42% of schools offering CS courses is almost doubled by states like Maryland and Arkansas, who respectively have 83% and 78% of their high schools offering CS courses. As it

stands, California not only needs more students interested in the Computer Science field, which this activity aims to address, but also needs more schools to offer CS courses to students. Students simply can't take classes that just don't exist.

**Related Work**:

Some of the main research for this project was teaching strategies and information of student learning. According to Daniel Willingham's book Why Don't Students Like School, a strategy that helps students remember information easier is "chunking". Essentially, students have an easier time retaining information when the information is broken down into easier to digest segments rather than trying to memorize all the info at once. Another aspect of a students education to consider is that a students ability to critical think can be improved with background knowledge. This makes sense as can be seen when a student is taking an exam, a question asks about providing an algorithm that solves a given task. The student will have a much easier time solving this question with some background knowledge, maybe they've seen similar problems in class previously and having previously seen a similar problem they can have an easier time using critical thinking skills to develop a solution.

As this activity is teaching programming to people who have no prior experience, it would be important to understand some of the issues that may arise in the thought process of these students. The Buggy Path to The Development of Programming Expertise, an article Roy D. Pea refers to three of these misconceptions as "bugs". The first known as the "parallelism bug" refers to a misconception that the computer can compute multiple lines at once. Another one of these bugs, the "intentionality bug", references when a student wrongly infers the function of a program from the instructions. The final bug, known as the "egocentrism bug", is seen when a student assumes the computer knows more about what it should do than it actually does, also assuming the computer can infer missing code like a human.

The National Center for Women & Technology (NCWIT) is an organization that focuses on correcting underrepresented communities in computing. NCWIT offers various programs that help students of all age groups to experience computing and is a very good inspiration for this

project. NCWIT also offers a page that reviews some of the "best practices for computer outreach programs." One of the items listed is to make the outreach program matter. One way to do this is by using meaningful content. As this is an introduction for students with little to no programming experience, it was determined that some of the most important concepts were some of the very basic building blocks of Python including strings, integers, print and input. These concepts will be important if they decide to continue their CS education and it would be very beneficial for them to see these concepts now. NCWIT also suggests incorporating student choice, as "nothing works better than letting students choose what they want to pursue." The activity allows students to come up with their own questions on any kind of trivia they like (within reason), as it was felt that it would give students more freedom while completing the activity. Another main topic this page discusses is to "Build student confidence and professional identity." One way to do this is by offering student-centered assessment, allowing the students to examine their own learning. This was addressed in the activity by allowing time for the students to share their quiz questions with their neighbors and allow them to answer each other's quizzes.

**Methodology**:

### Initial Concept & MADLib Game:

To help increase a students exposure to the field of computer science, the idea was to create an outreach activity that introduces some basic programming concepts to students that have had little to no prior programming experience. Examining California's K-12 Standards for CS showed that students in seventh and eighth grades (the main target demographic of this activity) should be able to create "clearly named variables that store data, and perform operations on their contents." Some of the main intended learning objectives for this activity would be for students to be able to define strings and integers which may be foreign concepts to them. The students should also be able to understand what variables are and some operations that can be done on them. Following the activity, students should also understand how to call functions like "print" and "input". To address the main standard and some of the learning objectives, the first iteration of the activity was a Mad-Lib program as it was felt that variables could be used to store user answers to mimic the blanks in a Mad-Lib where someone writes their answers. Students would also

need to print questions to the screen and prompt for user input that they would then use later in the program to print the whole sentence with the user's input included. To address these concepts a lesson plan to teach students using step by step instructions was developed. The lesson plan is structured as such:

1. Students are first introduced to the concepts of strings and integers, as these are very important and most of the activity will build on these concepts.
2. Then students are introduced to the syntax of the print function and are asked to practice implementing this function on their own.
3. Students are then given an explanation and some examples of variables in python and asked to save a message in a variable and print the contents of the variable.
4. There is a brief example of some basic string concatenation, as that is necessary for students to know when replicating the blanks found in a MadLib.
5. Students are then shown how the input function works and asked to input their name, save it to a variable and print it in a sentence using string concatenation.

The concept of chunking was used in the development of the plan, as students were given smaller bits of information at a time and then asked to practice them before moving on to new material. At the end of the activity worksheet students would be asked to put all of their knowledge together to create this larger MadLib program. The instructions for the MadLib program ask students to prompt user input using descriptors such as "noun", "verb", "adjective", "adverb", etc. They would then print their story line by line using the print command and would use the basic implementation of string concatenation to inter-splice the input words in the middle of their output strings.

## Initial Test:

This "Mad Lib" program activity was tested with two junior high students who volunteered to work through the activity on a Saturday morning. The two students had some minor experience with programming prior, but for the most part were learning these

concepts for the first time. One problem that was encountered was that the students had trouble running code in PythonTutor as a student could not rerun their program until a change was made to the code meaning that the students would have to type a character and then immediately delete it in order to register a "change" so their program could rerun. Another issue was that the pace was slightly too fast for them, as they had some trouble keeping up with the pace of the instruction and would ask for quick clarifications if they didn't quite catch what was just said.

## Alterations and Second Test:

Following the initial test run, a different activity was designed using the MadLib activity as a "skeleton" and based on some of the feedback received. The new concept for the activity was a "quiz" game that was designed to teach similar concepts to the "Mad Lib" such as strings, variables, printing and input. The quiz added some new concepts that weren't initially covered such as conditionals and booleans and while integers were discussed students did not have to use them in the original MadLib program. For the new concepts the worksheet includes brand new sections that:

1. Introduces students to the concept of boolean and what "True" and "False" mean in the context of programming in Python.
2. Teaches students the if-else block and how it should be set up, leaves the conditional statement blank to be filled in later.
3. Gives an example of a conditional using the sign "==" and how the block will run if the statement is either true or false. Emphasizes how "=" and "==" are different, "=" is for assignment, while "==" is for conditionals
4. Gives students starter code to create one generic question so that students have a basis when they implement their own questions later.

To avoid any similar IDE issues in any future run throughs, the main IDE was changed to PythonTutor's Code Visualization Tool and not its Live Programming Tool. The code visualization tool features clearly labeled buttons that show students where they should click to run their code and after running the code offers an "edit this code" button that takes students directly back to the edit screen. This also means that students no

longer have to type something just to immediately delete it, so that the IDE will rerun their program, simplifying this process for the students. For the second test-run with the same two middle school students, the pace of the activity was slowed down in an effort to help them digest the new information easier. One thing that was done in this test was asking them more questions like: “Did you understand that?” or “Do you have any questions?” at regular intervals in an effort to ensure that they were able to follow along with the instruction.

**Intervention:**

The activity was tested in an Algebra 1 class at Gabrielino High School in San Gabriel, California. A survey was conducted to gather some background information of the students including their grade level and experience level with computer programming. About 25 students attended, comprising 22 freshmen, approximately 90% of the class, with the rest of the class made up of sophomores. Of the students that participated, 16 of them (~64%) said that they had no previous computer programming experience of any kind, eight students responded that they had some basic programming knowledge and one person said they had done a lot of programming work on their own in the past. As most of the class has no previous experience, this was the perfect demographic of students for this activity.

The activity was completed during a 50 minute class period and that may have impacted the overall pace of the activity as some students expressed that they had some trouble following along with the instruction. Although a small number of students did not appear engaged in following the activity, most were active in following the instruction and working on each section of the activity. When students were released to do the final section on their own, most students quietly worked on it. It was emphasized earlier in the period that students should ask questions if they did not understand certain aspects of the activity, especially since it was most of the classes first time working with programming. Based on some of the questions asked it would seem that students had some trouble understanding how variables work in python, as students were confused why their user input answers were overwritten when using the same variable multiple times. This was interesting, as it was a clear example of the egocentrism bug discussed by Pea

from my research as the students felt that the computer could understand that they were trying to save each of the answers.

Following the activity, the survey asked the students some questions about their feelings about future endeavors in the computer science field. When asked if they would consider taking any computer science classes, twelve students said that they had some interest in that now, while three said that they were strongly considering taking classes. Ten of the students said that they would now at least consider getting a degree in computer science and four said they would strongly consider getting their degree in CS. Following the completion of the activity, the teacher said that one of the female participants told him that she had no previous programming experience and that she had a lot of fun participating in the activity and really wanted to do more programming in the future. This one comment shows that the activity achieved its goal for at least one person.

In the Spring Quarter, there was an opportunity to make improvements to this activity and present it in a workshop through Cal Poly's Kennedy Library. In preparation for this workshop presentation, some changes were made to the worksheet document based on some of the feedback given by the high school students. It was noticed that some students had a more difficult time understanding that when a variable is set equal to a value at two points, only the second value is kept. To address this a short concept check section was added that helps students to visualize this concept. Some students also expressed that they had a harder time following the instruction because of the large blocks of text on the worksheet. So these larger blocks were condensed down and was changed to a bullet point format in an effort to make it more readable for the students.

The workshop was presented on April 27th in Kennedy Library's Windows computer lab with an approximate length of 90 minutes. Although ten students signed up through the library's website only one student actually attended. Because of this, the workshop did not utilize instruction and instead the student was allowed to work through the worksheet on her own and ask questions if she needed an explanation on something. This was disappointing, as there was no way to test improvements in the actual instruction, and this workshop acted more like a

tutoring session, something that was previously done in CSC 313. However, the student did seem engaged in the activity and worked through the worksheet diligently.

**Reflection:**

Overall, I believe that the activity was a success in helping introduce students to some of the basic concepts of computer programming and potentially increased their interest in learning more about the field. Prior to the project, I wasn't entirely sure what I wanted to do after the completion of my degree, but this project has shown me how rewarding it can be to teach students some new things. I feel I have a better understanding of all that goes into the programming projects. All the research I completed during this process allowed me to better understand what kinds of topics should be covered with students new to programming, their potential thought processes and some teaching methods that can better help students digest new information.

If I were given more time to work on the project, I would attempt to present this activity with a group of students over a longer period of time. I feel that making it a multi day outreach program allows for me to go in depth with different concepts and design programs that highlight each of these topics. One thing I noticed from the classroom presentation was that some of the crafty students figured out that I gave them the full code for my provided example question and that they could simply copy and paste the code without putting any effort into thinking about the solution. In the future, I would change this example code into pseudocode so that students still have a guideline to follow when trying to write their implementation without simply giving them the answer right away. Also, as students seemed to struggle with the idea that variables in python are overwritten each time they are set equal to something, I would implement something along the lines of a concept check which would emphasize this fact to the students. The final change would be to the overall format of the worksheet documents as some students expressed that they felt lost and overwhelmed by the large blocks of text throughout the document. To address this feedback I feel that it would be better to use bullet points that can hopefully be easier for students to follow along with.

**References:**

California's k-12 Computer Science Standards. California Department of Education. PDF File

Willingham, Daniel T. *Why Don't Students Like School?: A Cognitive Scientist Answers Questions About How the Mind Works and What It Means for Your Classroom*. San Francisco, CA: Jossey-Bass, 2010. PDF File. 4 February 2022.

Scott, Allison. *Computer Science in California's Schools: An Analysis of Access, Enrollment, and Equity*. 17 June, 2019. PDF File. 5 February 2022.

Koshy, Sonia. The California Computer Science Access Report. 1 September, 2021. Web. https://www.kaporcenter.org/wp-content/uploads/2021/09/KC21007_CSCA_Access_Report.pdf Accessed 21 May, 2022.

Pea, Roy. *The Buggy Path to the Development of Programming Expertise*. January, 1987. Web. 4 April 2022.

Xie, Benjamin. *A theory of instruction for introductory programming skills*. 27 January, 2019. Web. 18 April, 2022. https://doi.org/10.1080/08993408.2019.1565235

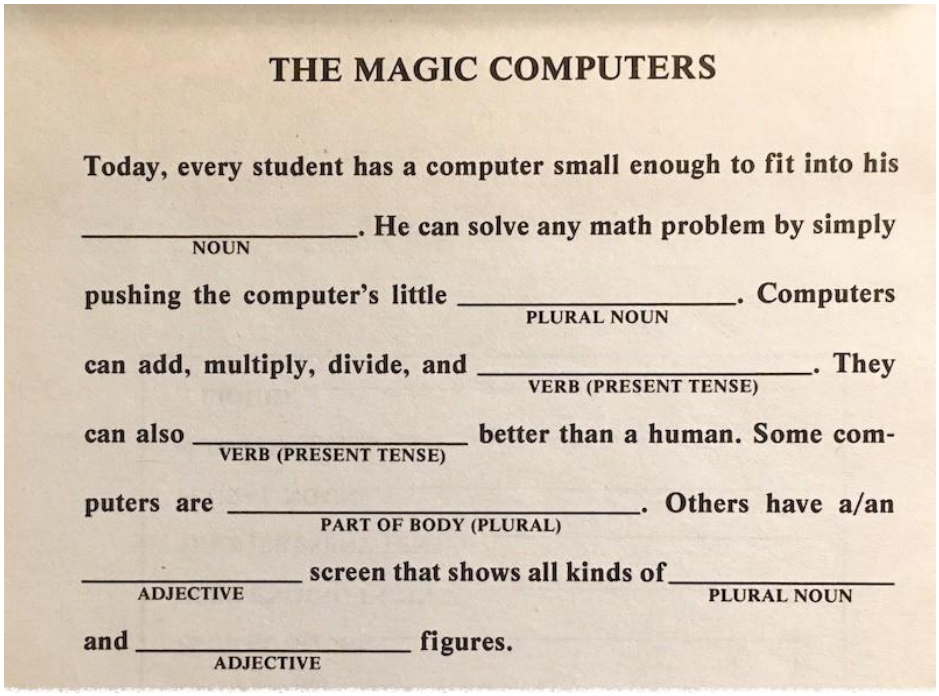
Krishnamurthi, Shriram. *Programming Paradigms and Beyond*. 2019. Web. 5 April, 2022. https://cs.brown.edu/~sk/Publications/Papers/Published/kf-prog-paradigms-and-beyond/paper.pdf

**Appendices:**

**Building a Mad Lib in Python**

What is a Mad Lib?

Mad Libs is a word game where one player prompts others for a list of words to substitute for blanks in a story before reading it aloud. An example is provided below.



**THE MAGIC COMPUTERS**

**Today, every student has a computer small enough to fit into his _____ (NOUN). He can solve any math problem by simply pushing the computer's little _____ (PLURAL NOUN). Computers can add, multiply, divide, and _____ (VERB (PRESENT TENSE)). They can also _____ (VERB (PRESENT TENSE)) better than a human. Some computers are _____ (PART OF BODY (PLURAL)). Others have a/an _____ (ADJECTIVE) screen that shows all kinds of _____ (PLURAL NOUN) and _____ (ADJECTIVE) figures.**

Computers represent data and information in different ways compared to humans. We think of things like words and sentences, while a computer represents these as something called **strings**.

The first thing we should understand is the concept of a **String**. A string is a representation of a sequence of characters. Some examples of Strings could be a

1. Single characters (“a”, “b” , “1”, etc)
2. Words (“Apple”, “computer”, etc)
3. Sentences (“Hello, World!”, “This is a String example”, etc).

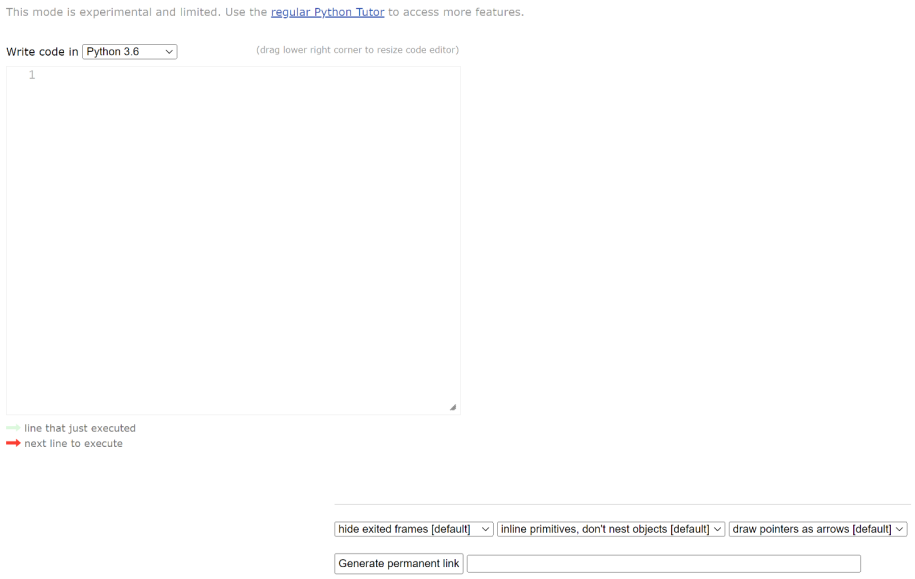
Now that we understand what a String is, let’s look at some commands that will act as building blocks for our Mad Lib Program

To create our Mad Lib program, we will go to this webpage:
https://pythontutor.com/live.html#mode=edit

Python Tutor's Live Programming Mode allows us to type our code and immediately see the output.

This is what you should be seeing on your screen. We'll type all our code into the **LARGE BOX** under "Write code in Python 3.6". The "1" shows us where the first line of the code will go.

Note: If you want to rerun your code use the "<< First" button at the bottom of the page to reset and then use "Next >" to run code line by line or "Last >>" to run the entire code.




☐ **1) The first thing we may want to do is display a String on our screen. To do this we want to use the print command:**

```
print("I'll display this on the screen")
```

```
I'll display this on the screen
```

**Variables**: Computers can assign various different values to variables. In some other computer languages like JAVA we would need to tell the computer the type of the

variable (string, integer, character), but Python is extremely convenient and can determine the type when we assign a value to it.

**2) We can also store any String that we want to save for later and display it on the screen like this:**

message = “I want to use this string later”

print(message)

```
I want to use this string later
```

Frames

```
Global frame
 message  "I want to use this string later"
```

**3) We can also combine two strings, which is called concatenation in Computer Science terms.**

print(“Hello World,”, “it’s such a nice day!”)

```
Hello World, it's such a nice day!
```

**Input**: In this program we want to simulate the blank spaces in our story. To achieve this we will use something provided by the programming language. Input allows us to print a string to the screen (like a question) and the user can type a response.

**4) Okay, we now know how to display String to the screen. But we want people to type their responses for our Mad Lib. We can do this by using the input command in Python.**

Value = input(“Enter your name: ”)

```
Enter your name: Alex
```

Frames Objects

```
Global frame
 value  "Alex"
```

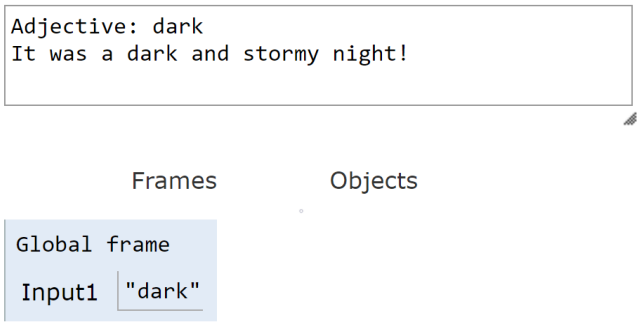
print(Value)

☐ **5) So, if we want to think about it terms of our Mad Lib our code should look something like this:**

Input1 = input(“Adjective: “)

☐ **6) The final piece for our Mad Lib puzzle is to insert the response to each question into our sentences. We can use the same format as when we concatenated the two strings to concatenate our user’s response.**

print(“It was a”, Input1, “and stormy night”)




Now that we have these blocks, we can build our Mad Lib program.Take the next few minutes to create your own Mad Lib game that has 3 lines and 3 inputs. Take in all the inputs first and then print out your story. **Remember that a Mad Lib is a story and has a beginning, middle and end. You must have at least 1 sentence where the input is in the middle of the sentence.**

***Keep in mind, you may be called to share your Mad Lib.***

***If you need some help thinking of certain words***

Nouns:

| | | | | | |
|---|---|---|---|---|---|
| apple | daughter | island | sidewalk | title | patch |
| badge | earthquake | judge | smoke | umbrella | playground |
| bubble | feast | lamp | bathtub | vegetable | mustang |
| bucket | frame | month | son | visitor | basket |
| cloth | ghost | ocean | stage | year | furniture |
| coach | giraffe | plane | throne | day | posion |

Verbs:

| | | | | | |
|---|---|---|---|---|---|
| reach | yell | fix | kick | open | allow |
| run | bake | grab | land | pass | end |
| stay | call | hang | lock | promise | fasten |
| talk | chase | imagine | mix | swim | gather |
| turn | drop | itch | notice | untie | jog |
| walk | escape | jump | obey | vanish | visit |

Adjectives:

| | | | | | |
|---|---|---|---|---|---|
| abundant | calm | hot | lazy | quick | thankful |
| angry | dirty | huge | light | quiet | vast |
| beautiful | dry | important | obnoxious | rich | warm |
| big | early | jealous | petite | scary | wet |

| | | | | | |
|---|---|---|---|---|---|
| brave | easy | kind | plain | short | worried |
| broken | empty | late | powerful | slow | young |

Adverbs:

| | | | | | |
|---|---|---|---|---|---|
| angrily | cheerfully | hastily | lazily | often | truly |
| anxiously | daintily | helpfully | lightly | partially | upbeat |
| beautifully | excitedly | honestly | mockingly | rudely | vainly |
| blindly | extremely | innocently | more | safely | warmly |
| bravely | freely | instantly | neatly | soon | wisely |
| calmly | generally | kindly | nervously | swiftly | wrongly |

**Making a Quiz Game in Python - A brief Intro to CS Concepts (Version 1)**

We will be using Python to create a quiz game in python that keeps track of how many correct answers the user has gotten.

Computers represent data and information in different ways compared to humans. We think of things like words and sentences, while a computer represents these as something called **strings**.

The first thing we should understand is the concept of a **String**. A string is a representation of a sequence of characters. Some examples of Strings could be a

1. Single characters (“a”, “b” , “1”, etc)
2. Words (“Apple”, “computer”, etc)
3. Sentences (“Hello, World!”, “This is a String example”, etc).

Another way that computers represent data is an **integer** which is one of the ways that a computer can represent a number. Keep in mind an **Integer** is always a whole number so it will never use a decimal point. Some examples of Integers can be 1, 34, 200, etc.

Now that we understand what a String and Integer are, let’s look at some commands that will help us to create our quiz

To create our quiz program, we will go to this webpage: Python Tutor

Python Tutor’s Code Visualization Mode allows us to type our code and immediately see the output.

This is what you should be seeing on your screen. We’ll type all our code into the **LARGE BOX** under “Write code in Python 3.6”. The “1” shows us where the first line of the code will go.

When you’re ready to run the code, hit the “Visualize Execution” button. You can run the code line-by-line using the “Next>” button or run all the code by using the “Last>>” button.

When you want to make changes to your code, click “Edit this code” which will take you back to the edit screen

1) **The first thing we may want to do is display a String on our screen. To do this we will use the idea of "printing". Most people think of printing as using a printer and putting text or images on a piece of paper.**

```
print("I'll display this on the screen")
```

```
I'll display this on the screen
```

2) **If we want to display a number we can use the same print function, with the only change being that we can replace the string with an integer.**

```
print(100)
```

**Variables**: Computers can assign various different values to variables. In some other computer languages like JAVA we would need to tell the computer the type of the variable (string, integer, character), but Python is extremely convenient and can determine the type when we assign a value to it.

3) **We can also store any String that we want to save for later and display it on the screen like this:**

```
message = "I want to use this string later"
print(message)
```

```
I want to use this string later
```

Frames

```
Global frame
message | "I want to use this string later"
```

4) **We can also use variables to save any integers for later usage, we do the exact same thing except we set the variable equal to a integer instead:**

number = 15

5) **If we want to add 2 to our integer variable, we could do so as follows:**

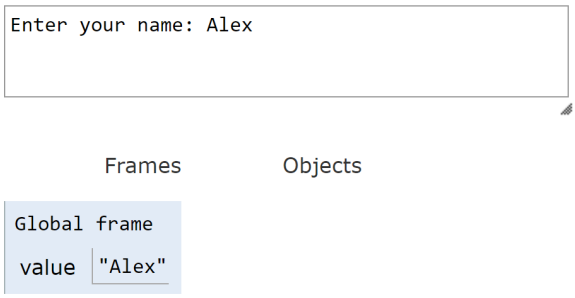
number = number + 2

What will the new number be?

**Input**: In this program we want to allow the user to answer our questions. To achieve this we will use something provided by the programming language. Input allows us to print a string to the screen (like a question) and the user can type a response.

6) **Okay, we now know how to display Strings to the screen. But we want people to type their responses for our quiz. We can do this by using the input command in Python. Let's try a basic input asking someone for their name**

name = input("Enter your name: ")




***** If we look at the "Global frame" box, we can see that the name Alex was saved in the variable called "name" *****

**Note: When running the program with inputs, the "Last>>" button will go up until the next input. Once you "submit" your input, you will need to hit either "Next" or "Last" again.**

**Booleans**: In the program we want to check if the user's answer matches our correct answer, so we will use a computer data type called a boolean. Simply a boolean is a value that tells us when a value is **true** or **false**. In this program we will use this for equality, meaning that if two values are equal to each other the "true" boolean is used.

☐ **7) To check our users answers we can use a code block referred to as if-else that will help us to check for correctness. The code will look something like this:**

```
if (_____):
   print("Yes, that's true")
else:
   print("No, that's false")
```

☐ **8) We'll need to replace the Blank, so we'll have to use an equal check. For this we will need to use two equals signs, ==, since one equal sign is used when assigning variables.**

**For example, let's say that the user's answer was saved in Ans1 and the correct answer is "b".**

```
if (Ans1 == b)
```

This means that if the user answer matches the correct answer the boolean value will come out to equal true and we will go into the first branch that tells the user their answer was correct, and if the two DO NOT match the boolean will be false and will go into the else branch, that lets the user know they are wrong

Now let's put all this knowledge together and create our first question

**Let's write this question:**

Who is Mario's brother?

a) Wario
b) Tony
c) Luigi

**It will end up looking something like this:**

Note: The line denoted with # are comments to help understand what the line below does.

```
# score is set to zero
score = 0
# lets ask the user our first question
print("Who is Mario's brother?  ")
print("a) Wario")
print("b) Tony")
print("c) Luigi")
# here's where the user will input their answer to question 1
ans1 = input()

# questions 2 and 3 go here later

if (ans1 == "c"):
   # we add 1 to score when the user gets an answer correct
   score = score + 1
else:
    # we keep the score the same when the user gets the answer wrong
    score = score
# check the answers for Q's 2 and 3 here later

# let's show the user how many they got correct
print("You got",score,"answers correct out of 3")
```

Now, take some time to make your own quiz. Change the first question to something new and create 2 new questions. Once everyone is ready, you'll have your neighbors try and answer your questions and some will have the opportunity to share their quizzes with the class.

Please fill out this survey, when you are done. This will let me know what you liked and didn't like about the activity and will help me to improve the activity in the future.

**Thank you for your participation!**

**Writing a Trivia Quiz Game using Python Programming (Version 2)**

We will be using Python to create a quiz game in python that keeps track of how many correct answers the user has gotten.

This workshop will go over these Learning Objectives:

- Understand basics of Strings and Integers
- Understand how to print to the screen.
- Understand variables and how they work in Python
- Understand how to prompt for user input and store the result
- Understand how to do basic String Concatenation

Computers represent data and information in different ways compared to humans. We think of things like words and sentences, while a computer represents these as something called **strings**.

The first thing we should understand is the concept of a **String**. A string is a representation of a sequence of characters. Some examples of Strings could be a

1. Single characters (“a”, “b” , “1”, etc)
2. Words (“Apple”, “computer”, etc)
3. Sentences (“Hello, World!”, “This is a String example”, etc).

Another way that computers represent data is an **integer** which is one of the ways that a computer can represent a number. Keep in mind an **Integer** is **ALWAYS** a whole number so it will never use a decimal point. Some examples of Integers can be 1, 34, 200, etc.

Now that we understand what a String and Integer are, let’s look at some commands that will help us to create our quiz

- To create our quiz program, we will go to this webpage: Python Tutor

- We’ll type all our code into the **LARGE BOX** under “Write code in Python 3.6”. The “1” shows us where the first line of the code will go.

- When you’re ready to run the code, hit the “Visualize Execution” button. You can run the code line-by-line using the “Next>” button or run all the code by using the “Last>>” button.
- When you want to make changes to your code, click “Edit this code” which will take you back to the edit screen

1) **The first thing we may want to do is display a String on our screen. To do this we will use the method of "printing". Most people think of printing as using a printer and putting text or images on a piece of paper.**

```
print("I'll display this on the screen")
```

```
I'll display this on the screen
```

2) **If we want to display a number we can use the same print function, with the only change being that we can replace the string with an integer.**

```
print(100)
```

**Variables**: Computers can assign various different values to variables. In some other computer languages like JAVA we would need to tell the computer the type of the variable (string, integer, character), but Python is extremely convenient and can determine the type when we assign a value to it.

3) **We can also store any String that we want to save for later and display it on the screen like this:**

```
message = "I want to use this string later"
print(message)
```

```
I want to use this string later
```

Frames

```
Global frame
 message | "I want to use this string later"
```

4) **We can also use variables to save any integers for later usage, we do the exact same thing except we set the variable equal to a integer instead:**

number = 15

5) **If we want to add 2 to our integer variable, we could do so as follows:**

number = number + 2

## Concept Check

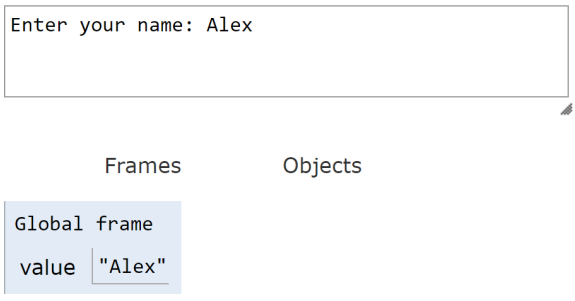
What will the new number be?

| number | |
|---|---|

**Input**: In this program we want to allow the user to answer our questions. To achieve this we will use something provided by the programming language. Input allows us to print a string to the screen (like a question) and the user can type a response.

6) **Okay, we now know how to display Strings to the screen. But we want people to type their responses for our quiz. We can do this by using the input command in Python. Let's try a basic input asking someone for their name**

value = input("Enter your name: ")



***** If we look at the "Global frame" box, we can see that the name Alex was saved in the variable called "value" *****

**Note: When running the program with inputs, the "Last>>" button will go up until the next input. Once you "submit" your input, you will need to hit either "Next" or "Last" again.**

**CONCEPT CHECK**

Show what will be stored in name after each line is run:

name = "Michael"
name = "Joe"

| name | |
|---|---|

**Booleans**: In the program we want to check if the user's answer matches our correct answer, so we will use a computer data type called a boolean. Simply a boolean is a value that tells us when a value is **true** or **false**. In this program we will use this for equality, meaning that if two values are equal to each other the "true" boolean is used.

☐ **7) To check our users answers we can use a code block referred to as if-else that will help us to check for correctness. The code will look something like this:**

```
if (_____):
    print("Yes, that's true")
else:
    print("No, that's false")
```

☐ **8) We'll need to replace the Blank, so we'll have to use an equal check. For this we will need to use two equals signs, ==, since one equal sign is used when assigning variables.**

**For example, let's say that the user's answer was saved in Ans1 and the correct answer is "b".**

```
if (Ans1 == b)
```

This means that:

- If it does match, it will go into the first branch
- If it does not match, it will go into the second branch

Now let's put all this knowledge together and create our first question

**Let's try to write this question:**

What statue is outside of the Baker Building?

a) Newton
b) Edison
c) Einstein

What is your answer?

**Here is the Pseudo-code (not actual code) to create this:**

- Create a variable to track score, set it to 0
- Print your question
    a. Print answer 1
    b. Print answer 2
    c. Print answer 3
- Create variable to keep track of user answer, use input to get the user's response

  Questions 2 and 3 will go here

- If the user answer equals the correct answer
    - Increase the users score by 1
- Otherwise
    - Leave the score the same
- Print the users final score out of the number of questions

Now take the time to create your quiz:

1. Change the first question to something new
2. Add two new questions
3. If you get stuck at any point, please ask questions

When you are done, please use this link: tinyurl.com/CSSurvey0504 to take a survey to answer questions about this workshop

**Thank you for your participation!**