**Course Module 1 – Python Programming**

\*\* I’ll score all of you in terms of the given problems being solved and your approaches for solving :p so try your best

\*\*For exercise’s sake, we’ll use the online python compiler/coding ground here <https://replit.com/languages/python3> (open an account)

\*Our course will focus on the meaning and reasoning behind the coding and syntax’ing conventions, (rather than just learning and memorizing syntaxes) and then using those understanding to do simple coding exercises. I’ll be teaching you variables, inputs and asking for inputs. functions and calling for functions (i.e., recursion) very early on, so that you can take multiple approaches towards solving a coding problem.

\*As we are focusing on conventions, we’ll use another programming language JavaScript to show the comparison between Python and other languages as well as the difference between compiled vs interpreted languages (highly important to understand input and output) {no need to remember or learn the JavaScript for this, just noticing the difference between the two would be enough}

\*\*Please look into the concepts I’ll be referring to in these resources for further understanding

Websites: <https://www.tutorialspoint.com/python/index.htm>   
<https://www.w3schools.com/python/default.asp>

Tutorial: [Foundations of Programming](https://drive.google.com/drive/folders/1b_AKPrtVVq1kCf32ybTnuuS6Is0fgQmG?usp=sharing) (Uses JavaScript)

Book: [Automate the Boring Stuff with Python](https://automatetheboringstuff.com/) (This will be mainly used for creative exercises)

Also, we’ll learn to use stack overflow, YouTube, reddit, GitHub and official documentation of a language so that we can find a solution for any problem by ourselves that we may stumble upon. (I’ll expect you to report what approaches you may have taken for a problem and where did you find it; there will be additional evaluation for it)

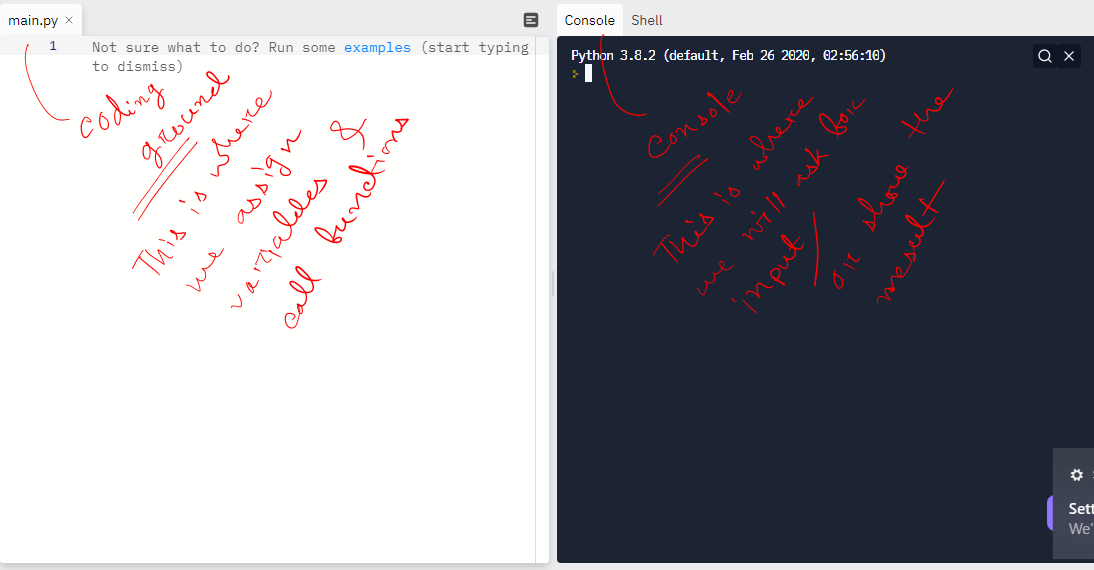
\*\*\*Download the images if they are hard to see

**Programming Interface**

First let’s get familiar with our programming interface (See photo below)

The coding ground is where we will write out code. The coding language is just instructions for the computer to run. The console is where those instructions get interpreted. We can make our program interactive by giving the console instructions to ask for user inputs (we will go in depth about that later), which will then be used to perform further instructions.

Example: You write a program that adds two numbers. Now there could be infinite two numbers and your computer have no way of knowing which two. So, what’s the most logical thing for a computer to do? To ask you which two numbers. But the computer can’t even do that unless you instruct them to ask you that question. So, you code it to ask you the numbers and when they get the numbers, to add them up. The console is the place where it executes your instructions, even the asking for two numbers part.



**Task: Describe some creative programs you can write using the console’s input asking functionality, doesn’t have to be mathematical, doesn’t even have to be ‘codable’. Just describe some possible system that can use this instruction-input-output functionality to do some great things (20 Points)**

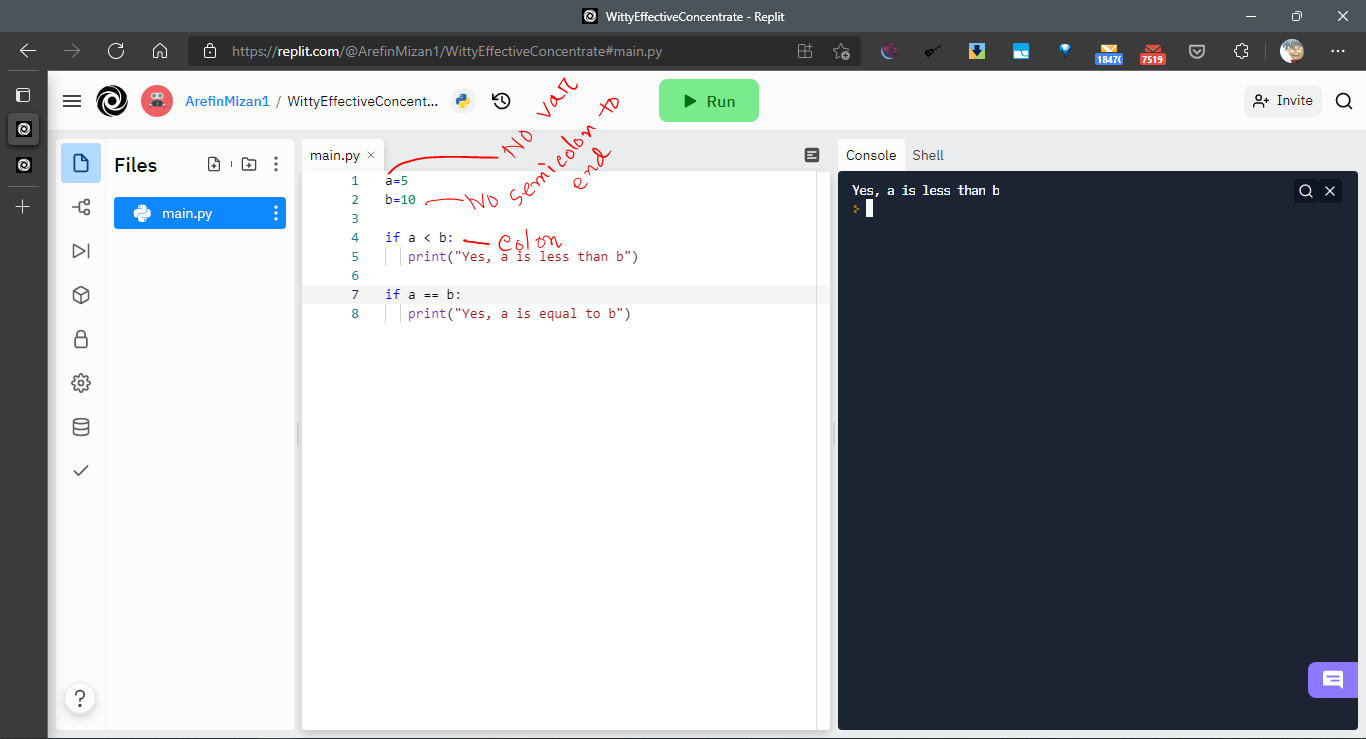
**Variables**

Below I used a conditional code (“if this, then that” program to be specific) in both Python and JavaScript (Fig 1 and 2) to describe what variables are. You can neglect the conditional part for now (it will be needed later\*\*), just focus on the first two lines of the code. We are assigning a value to ‘a’ and ‘b’ respectively. Assigning effectively means, the computer is storing this value in its memory, and you can retrieve that memory whenever you want for your usage. That’s why once you assign a variable to a particular ‘name’, here ‘a’ and ‘b’, these two can’t be used to assign any other variables in the same program. You can name a variable anything that you want and you can assign anything from numbers to words and lines to a variable for your usage.

For example, a could be assigned to a line ‘Abir is a genius hunk’; like this –

a = ”Abir is a genius hunk”

**(See that I use parenthesis for a line, this is something we will discuss in data types)**

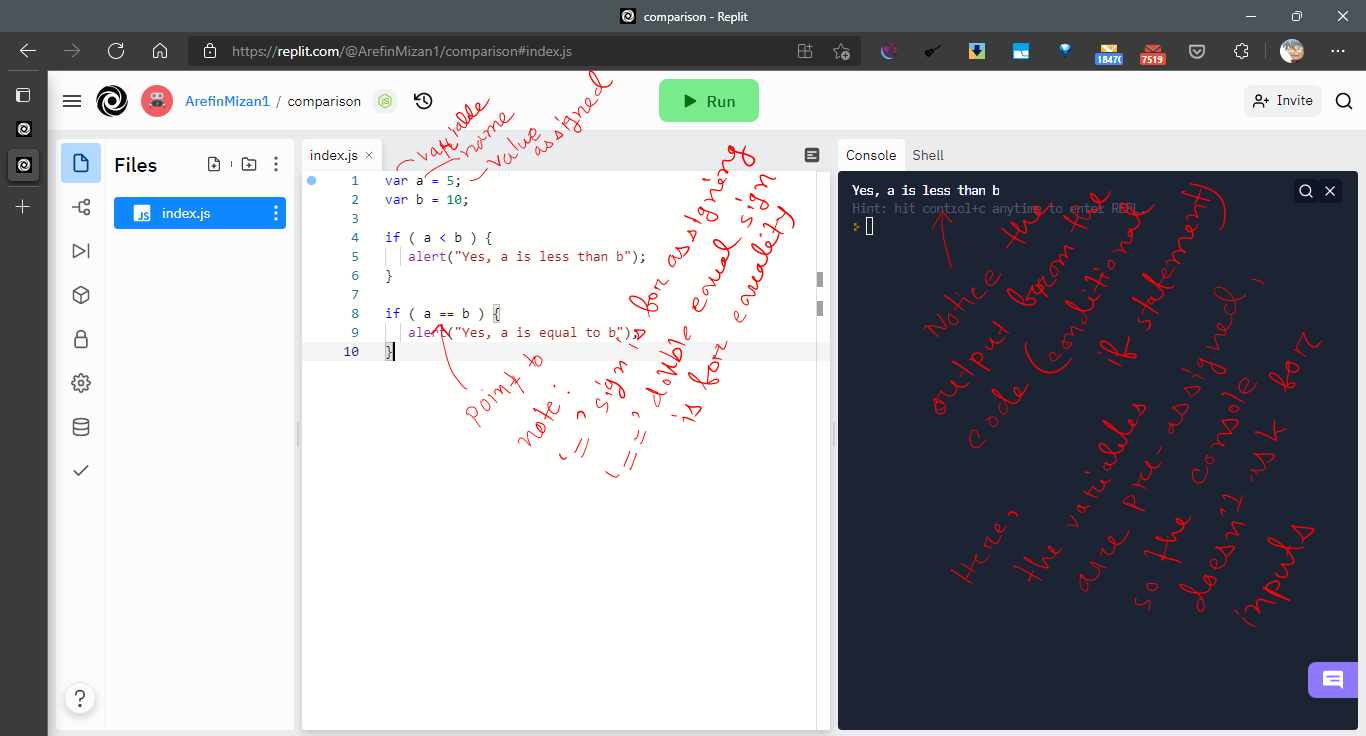
****

**Fig: 1 Python variable example**

Compare the two languages. There are different conventions of assigning variables, in other words ‘syntaxes’. In python, you can just use the ‘=’ sign after a name to assign a variable. In a language like the JavaScript you have to use the ‘var’ command before the name to assign a variable. Also, JavaScript is language derived from the C language (C++), so it requires you to use semicolon after the end of every statement/line. Python has no such requirements.

When you assign a value to a variable, the computer knows precisely what to work with. Unless you assign something to ‘a’ or ‘b’ the computer will make a variable but it will ask you to input what the value of the variables are (as we’ve discussed about inputs). That’s why assigning variables is a good way to work with data or calculations that is necessary for you. ***If you want a calculator, it’s better to not assign a value. If you want to find out the radius of a circle using a predefined value of pi, then it’s necessary to let the computer know the value of pi by assigning a variable named pi (or it could be any name that you want).***

Side notes: I’ve left other notes in the images that we will need later. You can google search them too if (I encourage you to). Like the python colon, or the double equal sign. Or the fact Python uses ‘print’ command while JavaScript uses the ‘alert’ command for same things.



**Fig 2: Javascript Variable Example**

**Task: Write down some creative ways you can use the conditional code in the above examples to achieve different things. Don’t have to write code, just describe the things you could use it for. (20 Marks)**

**Data Types**

There are 2 types of Data that you’ll be primarily working on as a beginner programmer. Text and numbers.

Texts in programming are called ‘Strings’. Any text units like words, lines and even paragraphs can be assigned as a string value. Funnily even numbers can be assigned a string value. We see numbers representing texts all around us, like a street number or a number plate. This is where numbers become strings.

Then there are numbers. And numbers are divided in two sub types (\*\*actually 3, but we only need two unless you are using complex numbers). The two sub-types are ‘Integers’ (পূর্ণসংখ্যা), or ‘Float’ (দশমিক).

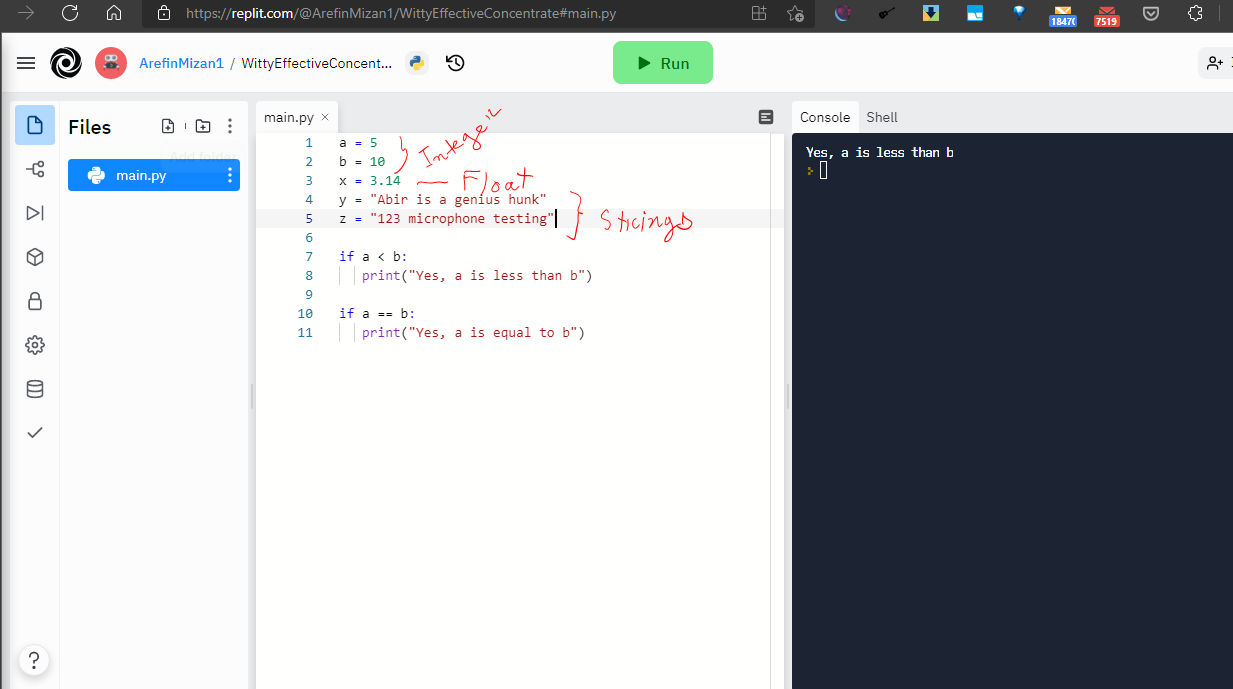
Ex. Integer: ……-3,-2,-1,0,1,2,3,…… (All the infinite numbers that can’t be represented as ভগ্নাংশ/দশমিক সংখ্যা)

Ex. Float: 0.2, 3.14 (pi) etc. (any number that can be presented as finite or infinite decimals)

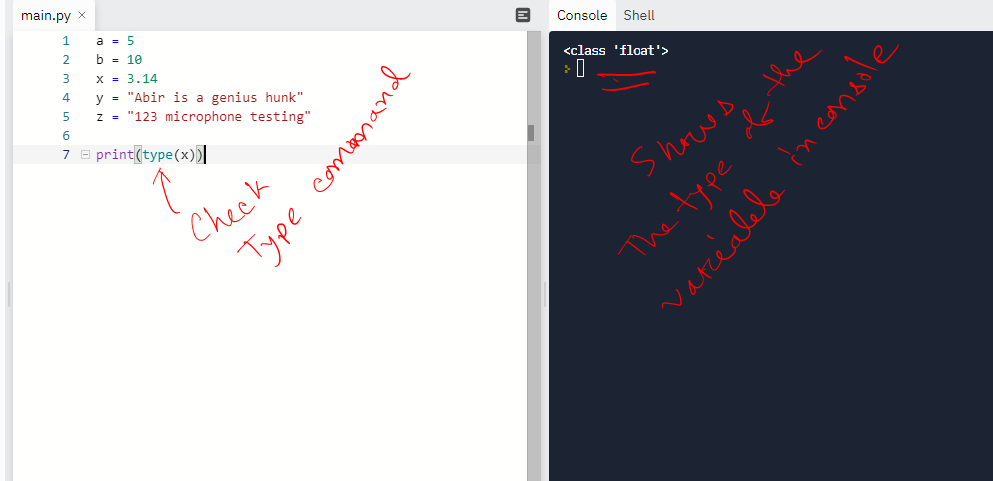
\*\*\* Differentiate between variable names and string data types always when writing code. (Think about it yourself).

\*\*\*A thing to remember, naming a variable has some strict conventions. You can’t use a number to name a variable, or any name that starts with a number. For example, 34, or 34abc can’t be the name of a variable.

\*\*\* A common variable naming convention is to use underscores ‘\_’. Examples: abc\_34

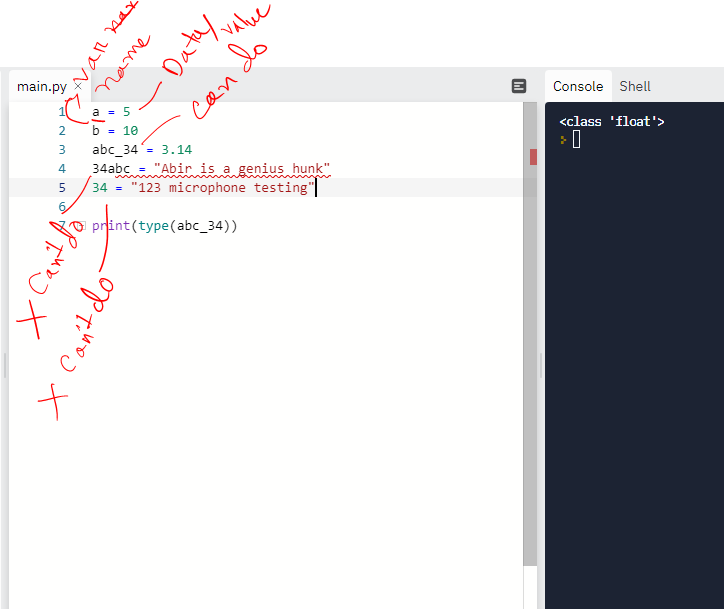


**Fig: 3 – Different Data types demonstrated**



**Fig: 4 – Manually Checking Data types demonstrated**

\*\* Notice that both the string variables are inside “ ” parenthesis. And this has to be maintained always. Python does not recognize strings if you don’t put them inside parenthesis. Inside parenthesis spaces, symbols and any characters are allowed.



**Fig 5: Variable and Data Types Naming Conventions**

Please read more about it in: <https://www.w3schools.com/python/python_datatypes.asp> (Ignore the other data types for now to not get overwhelmed)

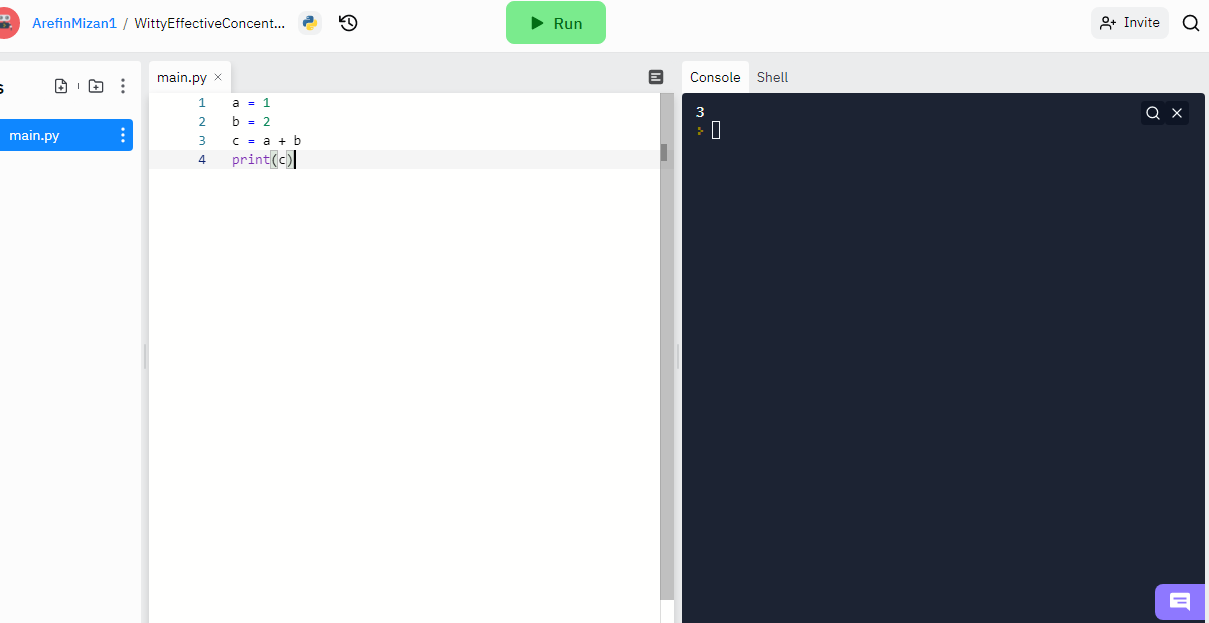
**Task: Think about how creatively you can name a variable and assign string values to it, so that in very large codebase (like 200-300 lines of code with multiple variables) you remember both the variable names and the string values {Use word based examples rather than lines or paragraphs} (20 Marks)**

**Input/Requesting Input**

We are about to enter the realm of actual ‘programming’. There will be less hand holding, more individual thinking (like a coder/problem solver). We’ve talked about inputs before. Almost all advanced programs are interactive, in other words, they require the user to input ‘secondary instructions’ before they can execute. Think about the software that you use, like an excel sheet. Before you can calculate your annual income, you have to ‘input’ your monthly income in a very regulated column, right? That’s your input. We as programmers use the console to input, because we don’t have a ‘graphical user interface’ like the excel or other software, and instead use a ‘command line interface’. If you write a program that adds two numbers, you’ll enter those two numbers in a black screened console, not in some beautiful user interface where you’ll have buttons to press.

Simple operation like adding two numbers don’t require specifying ‘data types’ that we’ve discussed before.

**Notice the difference:**

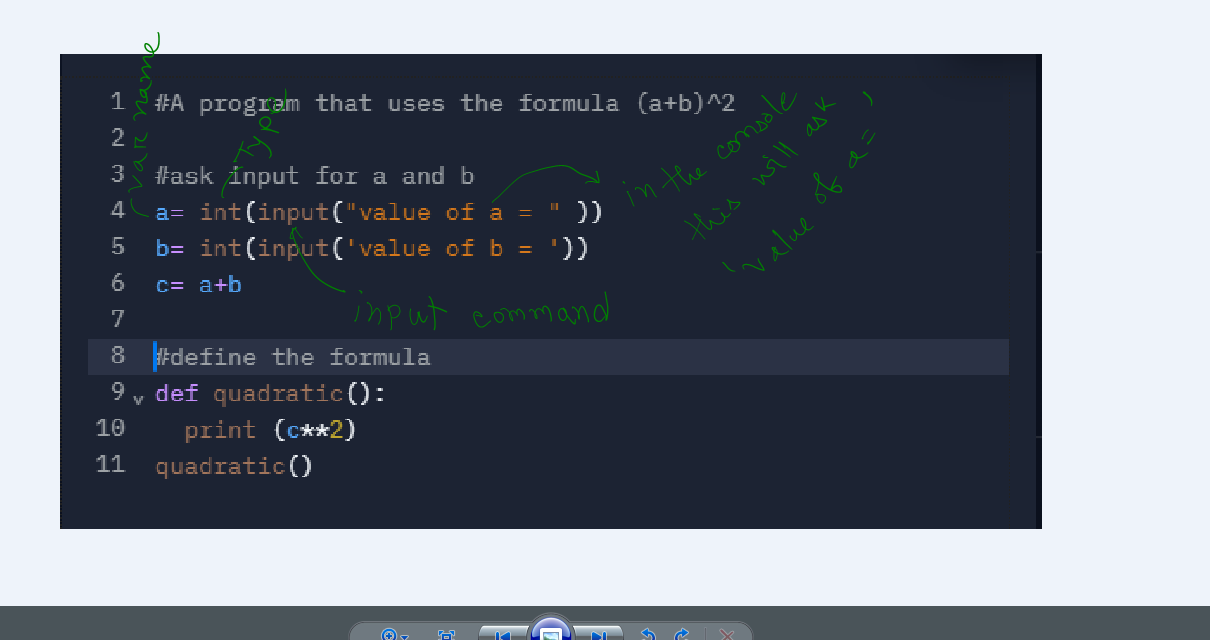
****

**Fig 6: See that the console didn’t ask for an input from the user because we’ve already established in our program that the two value we want to add is 1 and 2. Unless we change the values for a and b inside the code itself, no matter what operation you do with a and be (subtract, multiply, divide, exponentiation, permutation-combination; you name it), it will always return results pertaining to the assigned value.**

So, how do we direct the program to ask for inputs in the console?

We use the ‘input’ command. I’ll show you below.

Before I show it to you, one thing you should keep in mind, unlike manual assigning of values into variables, input variables require specifying ‘types’ (data types that we’ve discussed), which have their own commands (‘int’ for integers, ‘str’ for strings, ‘float’ for floats).



**Fig: 7 – A program that churns out the results for any ‘integer’ value of a or b that you input for a very well known and beloved formula (a+b)^2. {Run the program on the compiler and see for yourself}**

\*\*Inside the ‘input’ command brackets you can beautify the nature of the input as seen in the fig 7. Instead of just mindlessly waiting for an input in the console, the console will show a string “value of a =” and after the = sign you’ll enter the value. If you leave that bracket blank, it will still ask input, but the console won’t say anything, you’ll just input it. Try it both ways (Task)

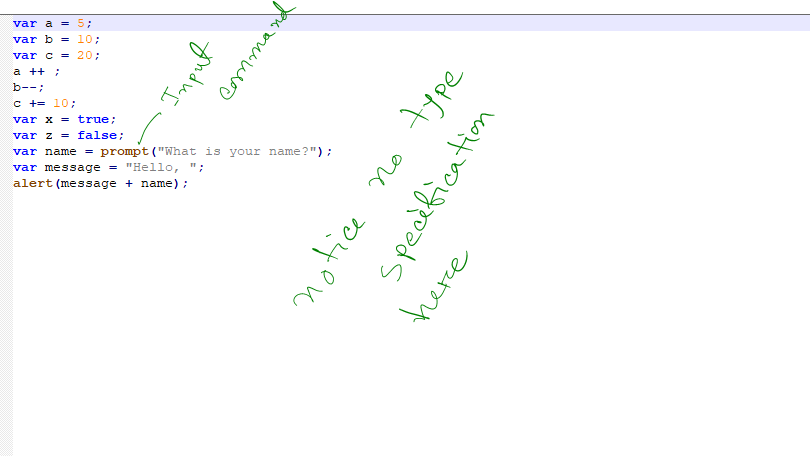


Fig 8: An example of input asking in JavaScript. See that it uses ‘prompt’ command for inputs. And there’s no type specification here. The reason is briefly explained below. Read it if you want, or don’t it’s not that necessary.

Python is what you called a statically typed or strongly typed language. Statically typed is a programming language characteristic in which variable types are explicitly declared and thus are determined at compile time.

Where as JavaScript is a dynamically typed language that can determine the variable type during runtime, that’s why type specification isn’t necessary (there are associated hassles and risks that comes with it, JavaScript is notorious for errors, you may think what types of risks can occur with dynamically typed languages\*\*).

\*\*\*Python can be dynamically typed too. But it’s a bad habit that you need not practice from early on. Always specify types.

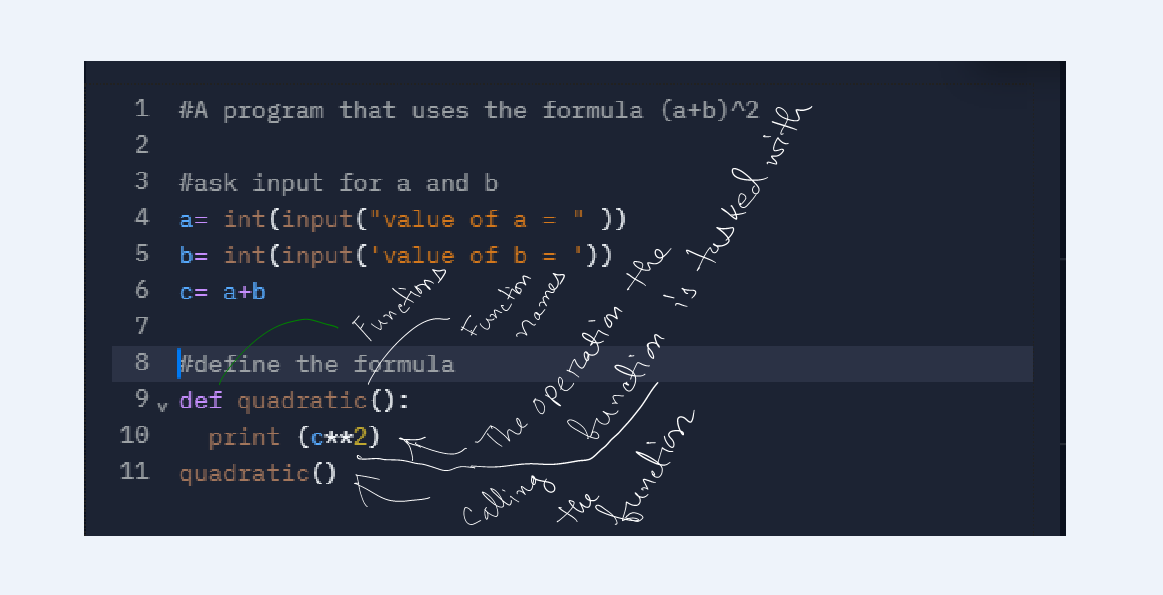
**Task: Write code that will ask string and float inputs. Think about what operations you can do with strings and floats, especially strings. Hint: you can ‘concatenate’. Find out other string-based operations and try to write input codes based on them. (20 Marks)**

**Functions**

Functions are the bread and butter of programming. This is what differentiates a ‘program’ from a ‘code’. The operational definition of a function is, A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.

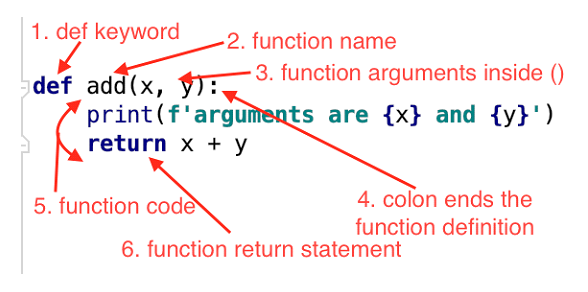
But forget all that mumbo-jumbo. In simpler words, a function is much like a variable in its nature, apart from the fact that it does things. It has a little bit of its own mind, in a sense that if you can give a generalized instruction, it can perform a set of tasks on its own. Functions are where programmers automate stuff.

Functions are established with the command ‘def’. Much like a variable, it needs to be named, look at the figure 9, where I named it ‘quadratic’. Then inside the brackets, you can either set the parameter (much like specifying types), or you can just get on to writing conditions or operations for the functions to perform. See the program in figure 9, it’s telling the function to exponentiate c with the power of 2 (the operator for exponents in python is ‘\*\*’)

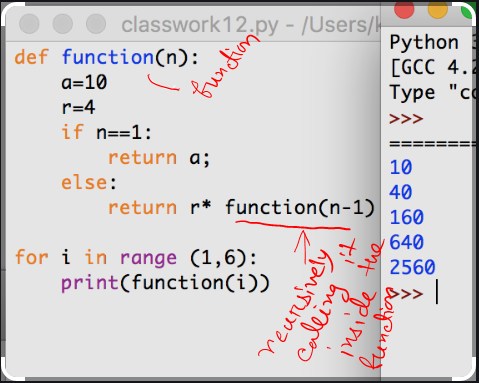


**Fig 9: Functions and all the pizzaz that comes with it using the same program that we used for inputs.**

Then we call a function whenever necessary by just simply writing its name. If you want to execute a function, then calling it is absolutely necessary. And also, notice that after naming a function you’ll need to use a colon ‘:’ to mark the end of the definition.

****

**\*\*A small token knowledge about a fascinating thing called recursion, which is calling a function inside the same function. For iterative codes that keeps looping until the desired result is reached (for loops, while loops), recursion saves a lot of space and lines for the coder. We will need to learn recursion to solve complex programming problems, but it’s still too advanced to discuss here. But I encourage you to google search or see tutorials of it anyway.**

****

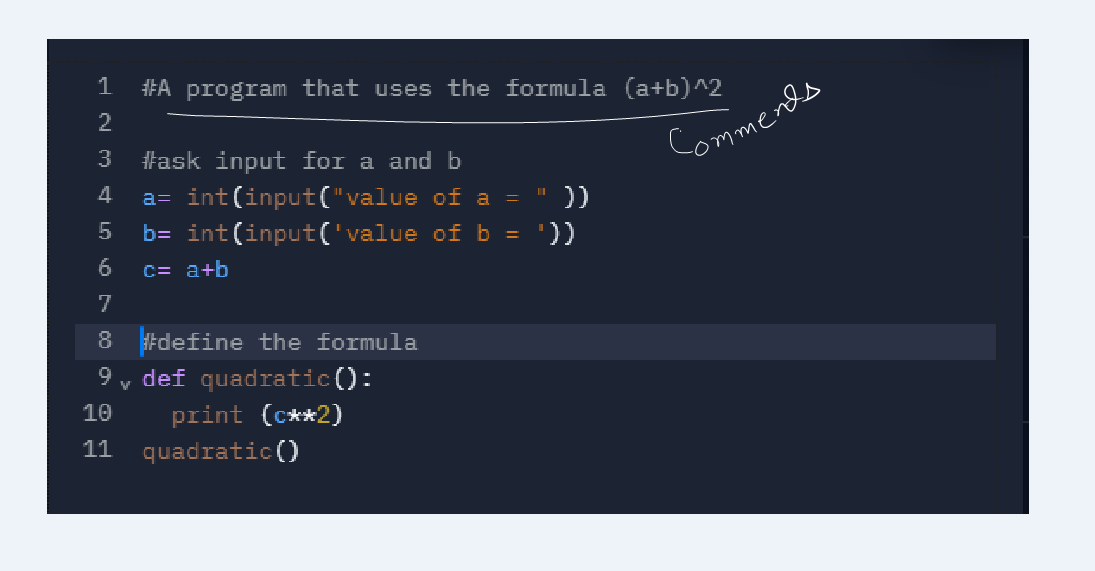
**Task:**

**1. Using the (a+b)^2 program as an example, write a function for cubing.**

**2. Write a program which accepts the radius of a circle from the user and computes the area with pi = 3.14 {use the formula for calculating radius given pi and area}(hint: use the input-function to insert the radius and convert it with the function float(radius) to a float).**

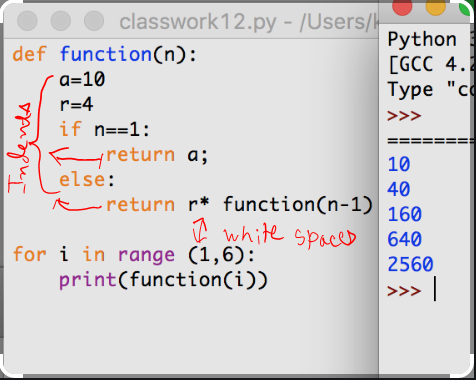
**Total: 20 Marks**

**Comments, Indents and White Spaces**

****

See the above image. I used 3 lines in line 1, 3 and 8 using a hashtag ‘#’, denoting the nature of the following actions I’ll take in the code. These are called comments and they mean nothing. They are a programming etiquette. Before you do an operation, you describe what you are doing or what this block of code is supposed to do with comments. Even if you don’t write them, your code will work fine, but commenting not only makes the code look cleaner, they make it easier for whoever’s using it.

Always make it a habit to comment in your code. And you can use comment wherever you want, even beside an actual line of code, using a hashtag or ‘//’ double slashes. The console or compiler will not regard it as code because of the hashtags/double slashes, that’s why they can be used anywhere in the code. But for cleanliness, we make it a habit to use them in separate lines.

****

Now see this image, There’s a ‘paragraph’ like space after the function and after the if statement and the else statement. This empty space is called ‘indentation’. Indentation means the following line of code are under a specific ‘group’ of codes, in the first case they are under the function, in the second case it’s under the if statement and in the third case under the else statement. This grouping can work infinitely, some coder’s codebase looks extremely wide because they group things so much. To avoid fattening your code, there’s something called object-oriented programming. But that’s outside of our scope for now. I still encourage you to read up about object orientation.

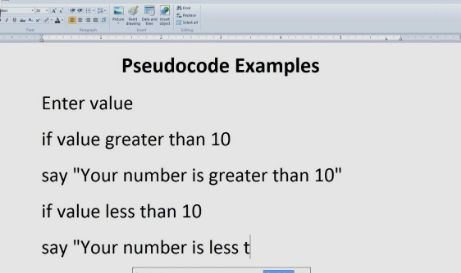
\*\*If you don’t use indents, your code will give you an error. And this is the foremost cases where people make mistakes, forgetting indents. Always be watchful about this.

\*\*\*Just like a function, indents are always ended with a colon ‘:’, whether they are if/else statements or for/while statements.

White spaces on the other hand, are a single line space that you leave empty after ‘grouping’ a block of code so as to not accidentally make the following code part of that group. See the for loop in the above image leaves exactly one line space after that indented group under the function. White spaces can be two or more lines as well depending on your artistic views.

**Task: There’s no particular task for this part, but I’ll mark your previous tasks based on your commenting, indenting and white spacing etiquette as well.**

**Pseudocode & Algorithm**



Pseudocodes should be self-explanatory from the above image. It’s more of a bridge between the English language and the programming language. Pseudocodes are a great tool for beginner and advanced coder alike and a core part of the process of ‘thinking like a coder’. You can’t formulate a problem in front of you and simultaneously code it down, because the programming language is not as intuitive to us as the English language or even our own language is.

So pseudocodes are in essences, writing down the individual steps of the problem in your most comfortable way, but in a program like fashion. For example, the above image makes a bad habit of not indenting, but you can indent on your pseudocode, which will make translating it to actual code much easier on you. An advanced programmer’s pseudocode may as well be indistinguishable from the actual coding language.

\*\* Make the habit of hand-writing pseudocode for a problem, before trying to code it out.

Algorithms are a much more common word to you I believe, and you probably know the definition very well. This is also the main step to start thinking like a coder. You see, a coder not only has to solve a problem, it has to teach the computer how to do it, and in the most efficient and creative manner possible. And to do that, he needs to mark out each individual step of the problem, to understand each and every step of the problem thoroughly and intuitively.

Take the (a+b)^2 program. This is very intuitive to us, that a non-coder does not second guess on the individual steps, he just uses the formula memorized in his mind. But the computer can’t do that, so can’t the coder. He has to understand each step of the problem in any of the myriads of ways the problem can be solved. Here I take two approaches, your task is to tell which one would be more time efficient, and your additional task is to code them both out. They are essentially the same program, with different algorithms.

**1st Approach:**

1. You input the two number (a,b)
2. You add the two number
3. You square the added result
4. Et voila

**2nd Approach:**

1. You input the two number (a,b)
2. You first define the well known formula of (a+b)^2 = a^2+2ab+b^2
3. You call the function
4. Et voila

Both the approach essentially has the same number of steps, but can you tell me, which one will take more time to execute?

**Thinking like a Coder**

I guess this part doesn’t need much to explain now. You encounter a problem, and you not just think about solving it, you also think about how to solve it efficiently. That’s what thinking like a coder entail. And I will grade your tasks based on how effectively you’ve learned to do exactly that. Best of luck!

**Additional Challenges:**

**1. Input an integer (n) and write a function that computes the value of n+nn+nnn**

**2. Write a program that converts centigrade to Fahrenheit**

**3. Write a program that calculates the area of a triangle**

**4. Write a program that greets you good morning (calling your name)**

**\*Marks for the Additional Challenges: 100 (25 each)**

**{send me screenshot for the codes and execution and for the creative thinking answers send answers in a word file}**