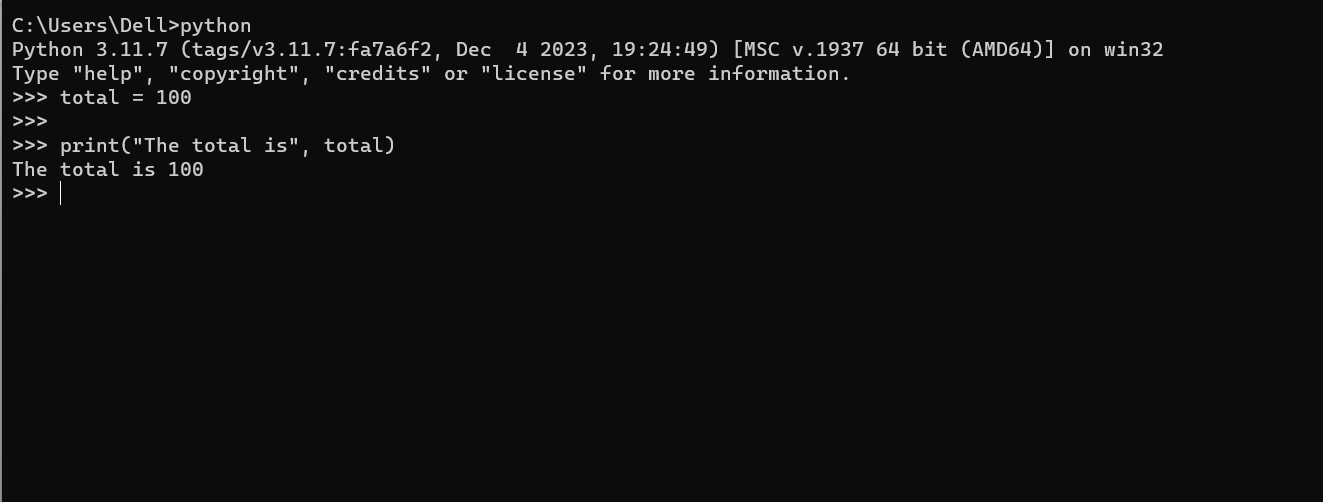
Week 2

Prithu Adhikari

**TASK**: Try inputting the following code and examine the results.

total = 100

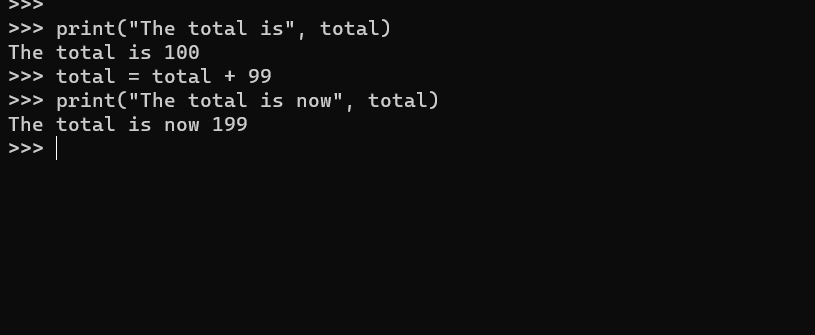
print("The total is", total)



**TASK**: Try inputting the following code and examine the results.

total = total + 99

print("The total is now", total)



**TASK**: Try inputting the following code, but replace each of the assignment expressions with the equivalent augmented assignment.

total = total - 1

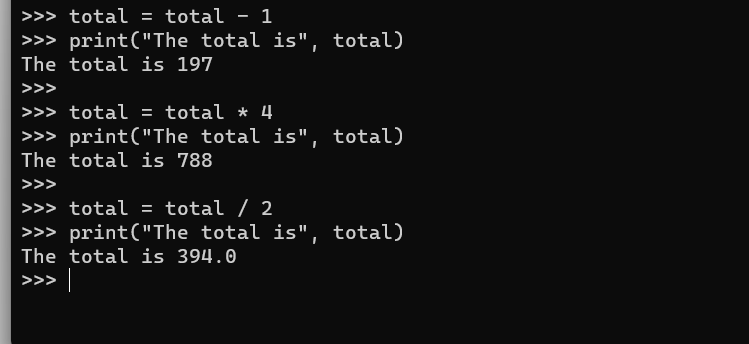
print("The total is", total)

total = total \* 4

print("The total is", total)

total = total / 2

print("The total is", total)

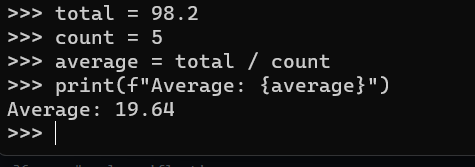


**TASK**: Try extending the code below so that it creates a new variable called ‘average’, that is set to equal the average calculated from the two other variables.

total = 98.2

count = 5

# add your extra code here



**TASK**: Use the type() *function* to determine the type of each of the following values.

False

1000

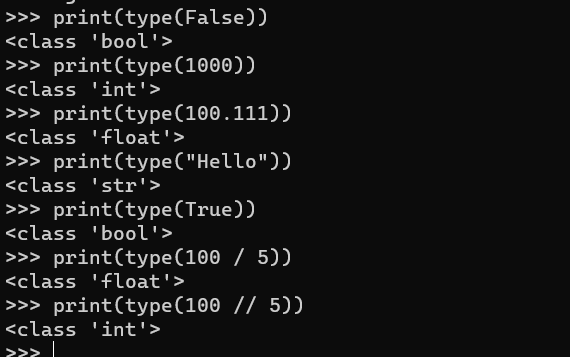
100.111

"Hello"

True

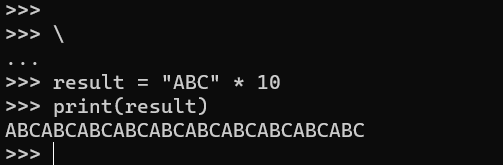
100 / 5

100 // 5

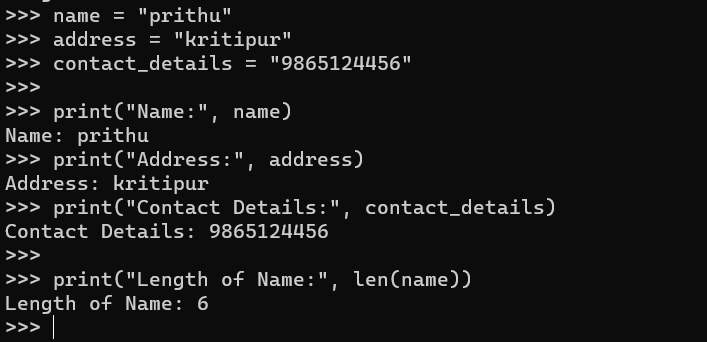


**TASK**: Input the following code and examine the result. What is the \* operator doing to the given string operand?

"ABC" \* 10



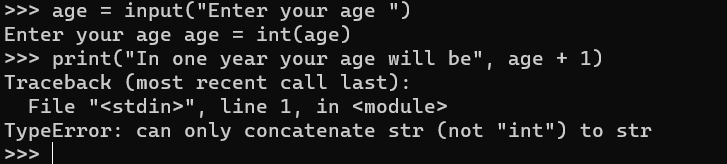
**TASK**: Write some code that calls the print() function several times, displaying your name, address and contact details. Add additional calls to the print() function which includes an argument that calculates and prints the length of your name, by calling the len() function.



**TASK**: Input the following code, when asked to type your age input a numeric value such as 20. Does this program work? If not, why?

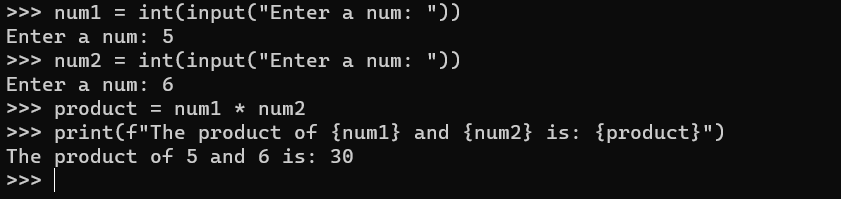
age = input("Enter your age ")

print("in one year your age will be", age + 1)



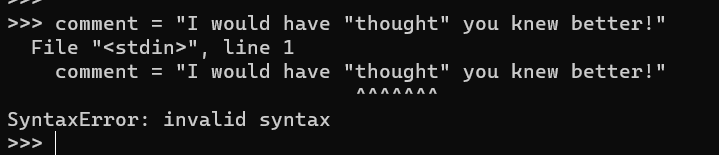
No, the provided code will not work as expected. The issue lies in the fact that the input() function returns a string, and attempting to concatenate a string with a number using the + operator will result in a TypeError

**TASK**: Write a program that prompts the user to input two numeric values. Once the values have been input display the product of these values, using the multiply (\*) operator.



comment = 'I would have "thought" you knew better!'

**TASK**: Try writing the above assignment statement but only use double quotes instead of single quotes as the string delimiter. What is the result?

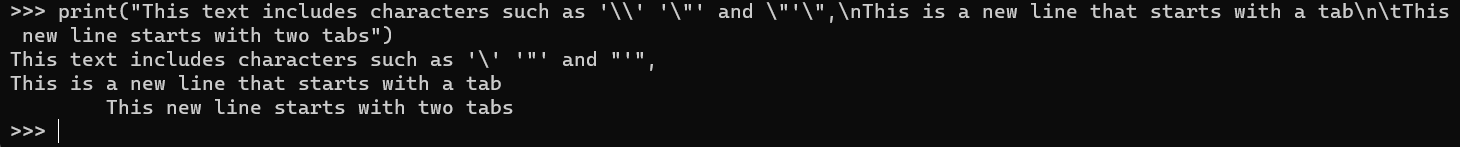


**TASK**: Write some code that calls a print() function, which takes a single string argument that results in the following text being displayed (exactly as shown).

This text includes characters such as '\' '"' and "'",

This is a new line that starts with a tab

This new line starts with two tabs



**TASK**: Write some code that calls a print() function, which takes a single string argument that results in the following text being displayed (exactly as shown).

\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

Hello there!

\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

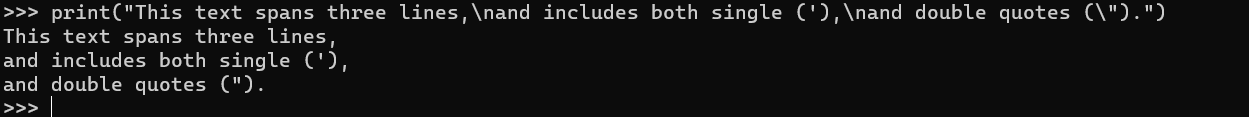


**TASK**: Write some code that calls a print() function, which takes a single string argument that results in the following text being displayed (exactly as shown). Do this without the use of any *escape sequences*.

This text spans three lines,

and includes both single ('),

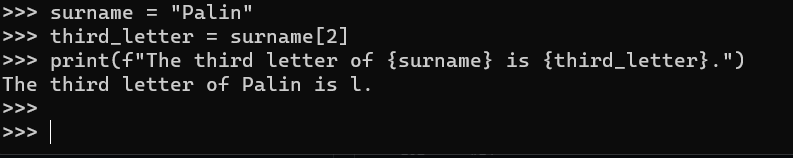
and double quotes (").



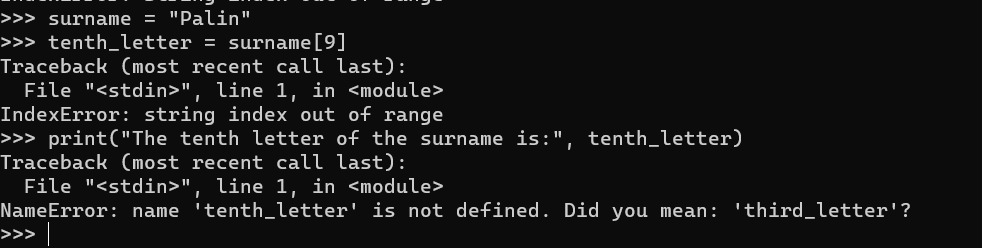
surname = "Palin"

initial = surname[0]

**TASK**: Rewrite the above example, so that the third letter of the ‘surname’ is accessed rather than the first, then print this letter to the screen.



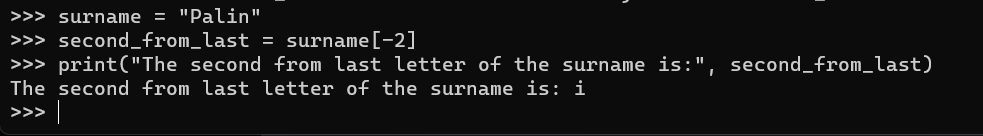
**TASK**: Rewrite the above example, so that the tenth letter of the ‘surname’ is accessed, and note the result.



surname = "Palin"

last = surname[-1]

**TASK**: Rewrite the above example, so that the second from last letter of the ‘surname’ is accessed rather than the last, then print this letter to the screen.

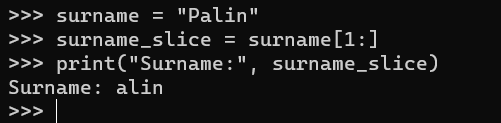


surname = "Palin"

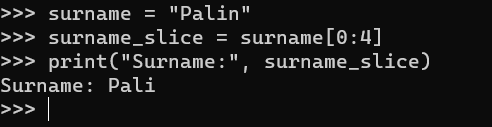
middle = surname[1:4]

Notice how the start indexed value is included, and the end indexed value is excluded.

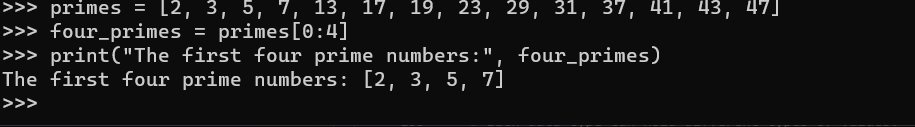
**TASK**: Rewrite the above example, so that all of the characters of the ‘surname’ except the *first* character are sliced and then displayed on the screen.



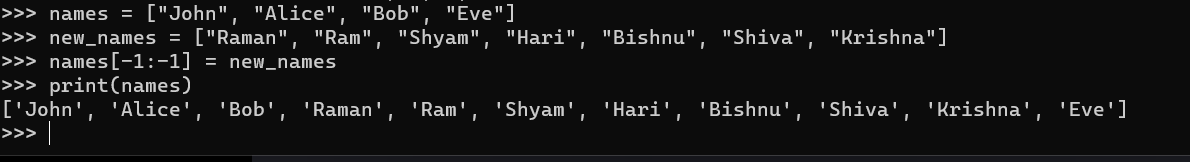
**TASK**: Write code that accesses and prints all characters of the ‘surname’ except the *last* character.



**TASK**: Write code that uses *slicing* to access then print the first four prime numbers defined within the ‘primes’ list given above. Note: you will have to input that list first for testing purposes.



**TASK**: Write code that uses *slicing* to insert two new names just before the final name within the ‘names’ list.



**TASK**: Work out in your head what the contents of the ‘nums’ list would be, then check this using the Python interpreter.

nums = [1,2,3] \* 5

print(nums)

#[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]

**TASK**: Look at each of the phrases below and ensure you understand what each of these means. For any that you do not understand, do a little research to find a definition of each term. This research may involve looking back over these notes, or the associated lecture notes. It may also involve searching for these terms on the Internet.

* Assignment
* Data-type
* Argument
* Indexing
* Slicing
* Mutable
* Immutable

Assignment: The process of giving a value to a variable. It involves storing a data value in a memory location identified by a variable name.

Data-type: A classification of the type of data that a variable can hold, such as integers, floating-point numbers, strings, etc.

Argument: A value that is passed to a function or method when it is called. Functions use arguments to perform their tasks or computations.

Indexing: The process of accessing individual elements in a sequence, like a list or string, using their position. In Python, indexing starts at 0.

Slicing: Extracting a portion of a sequence (like a list, string, or tuple) by specifying a range. It is a way to create a new sequence from parts of an existing one.

Mutable: Refers to objects whose value or state can be changed after creation. Lists in Python are mutable.

Immutable: Refers to objects whose value or state cannot be changed after creation. Strings and tuples in Python are immutable.