Intro to Coding

Class 3

Important Note

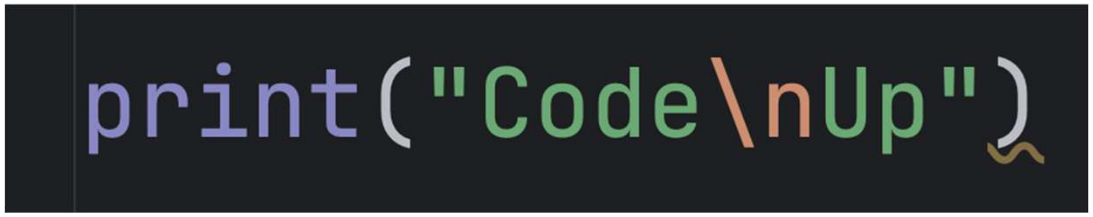
- **Do not delete your code** for the in-class activities. You will need it for a later activity!
    - You should still keep it after class as to have a reference for when you want to review
    - Create a folder with all your python files so you can find them later!

Strings Review

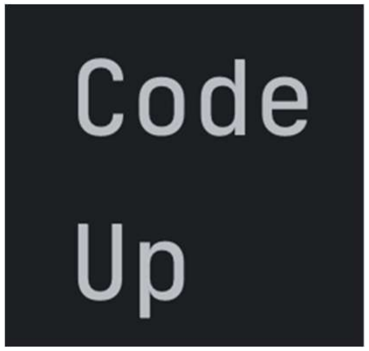
- A single variable that stores a text value (AKA it represents a word/sentence)
- Defined with quotation marks
    - Example: `name = “CodeUp”`

Using Backslash

- In various programming languages, backslash is used to "escape" certain characters
- Examples:
  - `\n` - creates a new line
  - `\t` - creates an indent
  - `\\` - prints the "\" character (putting "\" alone will create an error)
  - `\"` - prints a quotation mark character
- Can you think of why some characters might need to be "escaped"?



```
print("Code\nUp")
```



```
Code
Up
```

Backslash Activity

For this activity, you will be making your own quote! Print your quote and your name. It should follow this format.

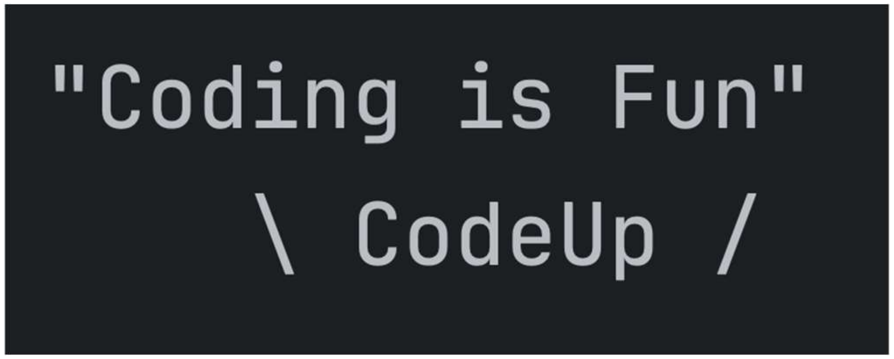
Bonus: modify your code to use only one print statement

“(your quote)”

\ Your Name /

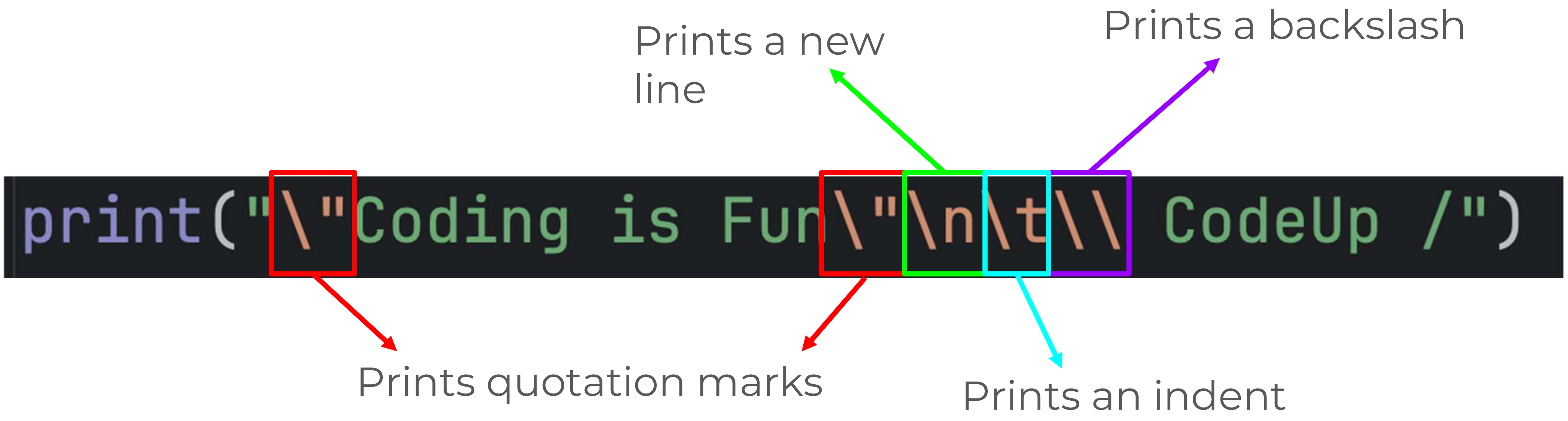
Note: you will be manually adding your name in the `print()` statement instead of taking it as input

Example



```
"Coding is Fun"
    \ CodeUp /
```

Solution

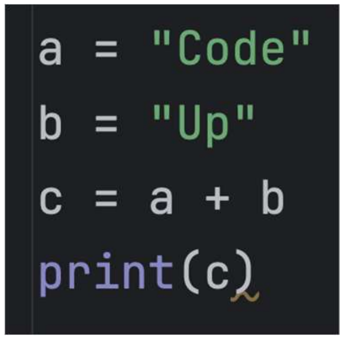


```python
print("\"Coding is Fun\"\n\t\\ CodeUp /")
```

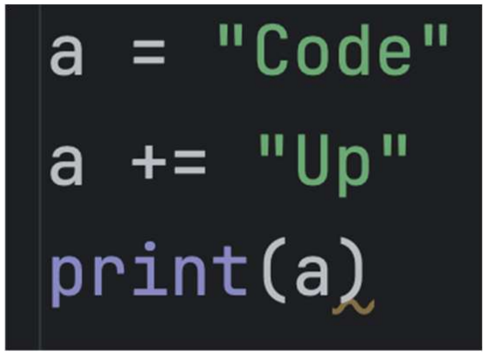
Review: Concatenating Strings

- Multiple strings can be concatenated (added) together
- String concatenation follows the <u>same format</u> as integer addition
  - `result = str1 + str2`
- If you want to add a string to the end of an existing string, you can do this
  - `str1 += str2`
  - `str1 += “a”` (adds the character “a” at the end of str1)

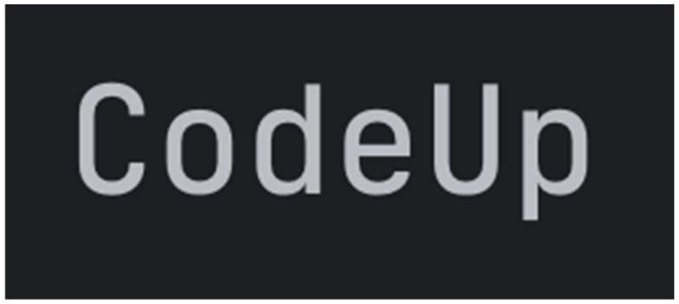
Both solutions lead to the same output →



```
a = "Code"
b = "Up"
c = a + b
print(c)
```



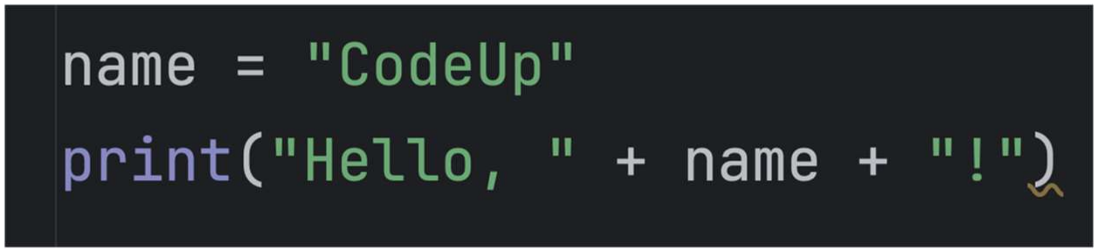
```
a = "Code"
a += "Up"
print(a)
```



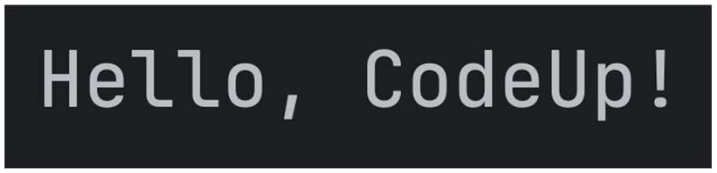
```
CodeUp
```

Review: Concatenating in a print() Statement

- The print statement supports concatenation
- You can directly concatenate within a print statement without storing it in a variable
- **Remember:** not all modifications need to be stored in an intermediary variable, oftentimes you can just make the modifications within the print statement
    - It's good practice to create only necessary variables for cleaner code



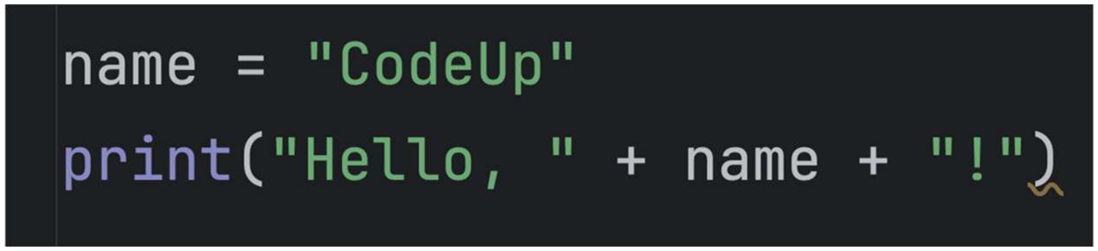
```python
name = "CodeUp"
print("Hello, " + name + "!")
```



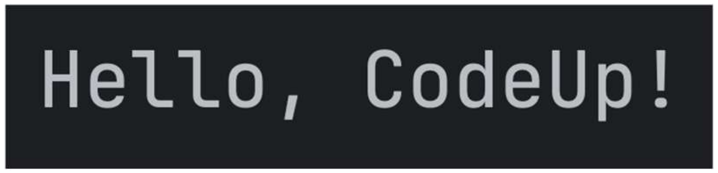
```
Hello, CodeUp!
```

Review: Concatenating Strings and Print Statements

- String concatenation follows the <u>same format</u> as integer addition
- Both of the following examples have the same resultant string!
  - `result = str1 + str2`
  - `str1 += str2`
- You can directly concatenate within a print statement without storing it in a variable
  - It's good practice to create only necessary variables for <u>cleaner code!!</u>



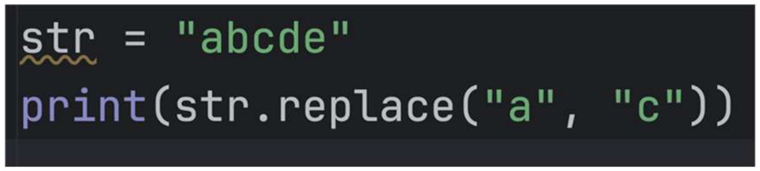
```python
name = "CodeUp"
print("Hello, " + name + "!")
```



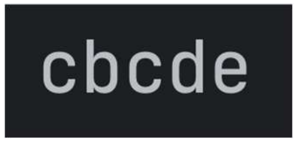
```
Hello, CodeUp!
```

String Functions

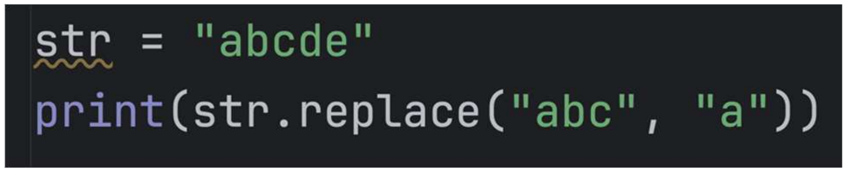
- `str.upper()` - makes the entire string uppercase
- `str.lower()` - makes the entire string lowercase
- `str.capitalize()` - makes the entire string lowercase **except** for the first letter
- `len(str)` - returns the number of characters in the string
- `str.replace(x, y)` - replaces all instances of `x` with `y`
    - **Note:** x and y are strings



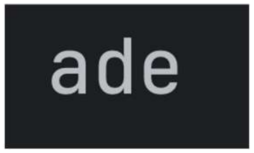
```
str = "abcde"
print(str.replace("a", "c"))
```



```
cbcde
```



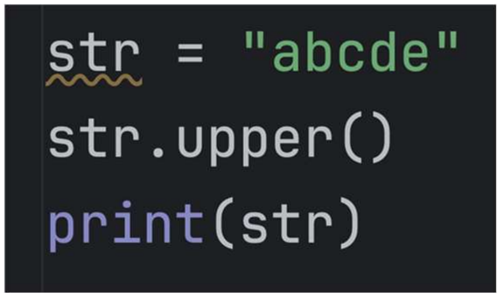
```
str = "abcde"
print(str.replace("abc", "a"))
```



```
ade
```

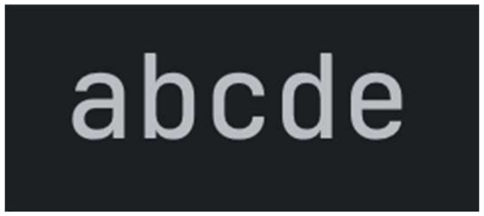
String Functions

Note: calling a string function on a blank line will not do anything to the string. This is because the upper function does **not modify the original string**. Rather, it **modifies a copy** of the original string.



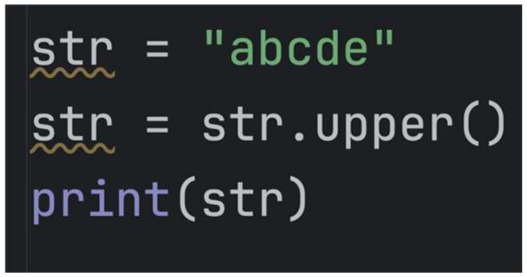
```
str = "abcde"
str.upper()
print(str)
```

Nothing changes

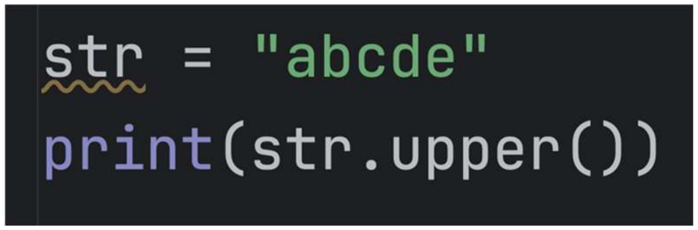


```
abcde
```

To avoid this, you can do either of the following:



```
str = "abcde"
str = str.upper()
print(str)
```

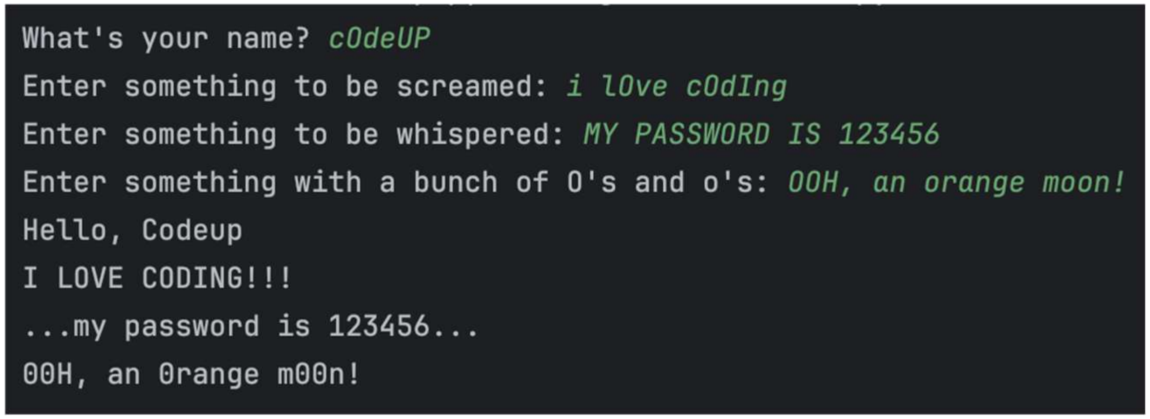


```
str = "abcde"
print(str.upper())
```

String Functions Activity

Write a program that takes in the following strings:

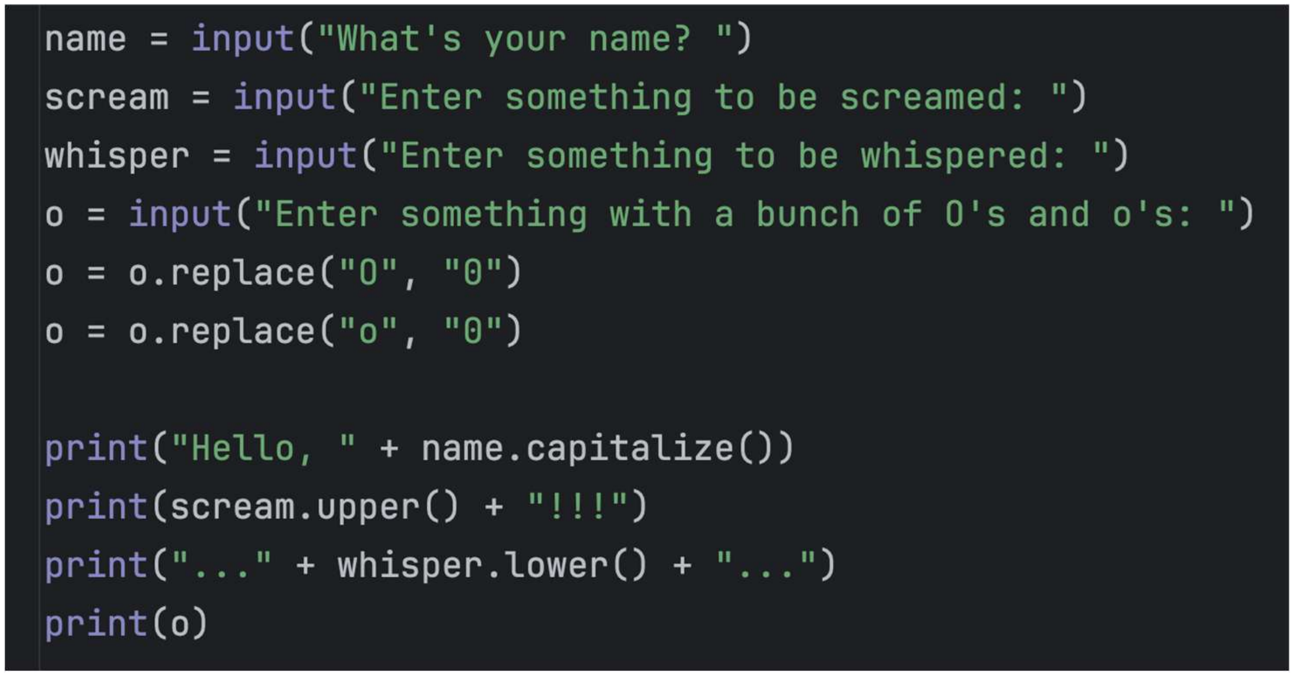
- Name string (just first name) → capitalize the first letter
- Scream string → append "!!!" at the end and capitalize all letters
- Whisper string → append "..." at the beginning and at the end and make all letters lowercase
- O string → replace all instances of "O" and "o" with the zero digit



```
What's your name? cOdeUP
Enter something to be screamed: i lOve cOdIng
Enter something to be whispered: MY PASSWORD IS 123456
Enter something with a bunch of O's and o's: OOH, an orange moon!
Hello, Codeup
I LOVE CODING!!!
...my password is 123456...
00H, an 0range m00n!
```

Solution

For the O string, simply run the replace function twice. Once for uppercase O, once for lowercase o.



```python
name = input("What's your name? ")
scream = input("Enter something to be screamed: ")
whisper = input("Enter something to be whispered: ")
o = input("Enter something with a bunch of O's and o's: ")
o = o.replace("O", "0")
o = o.replace("o", "0")

print("Hello, " + name.capitalize())
print(scream.upper() + "!!!")
print("..." + whisper.lower() + "...")
print(o)
```

Indexing

- To access individual characters within a string, we can do something called indexing
- In code, indices start at **0** and end at ***N* - 1**, inclusive, where *N* is the size of the string
  - **Note:** the first element is at index 0, not index 1

Index	`0`	`1`	`2`	`3`	`4`	`5`
Character	`C`	`o`	`d`	`e`	`U`	`p`

- Example: `print(str[0])` will print “`C`”, as that is the first character

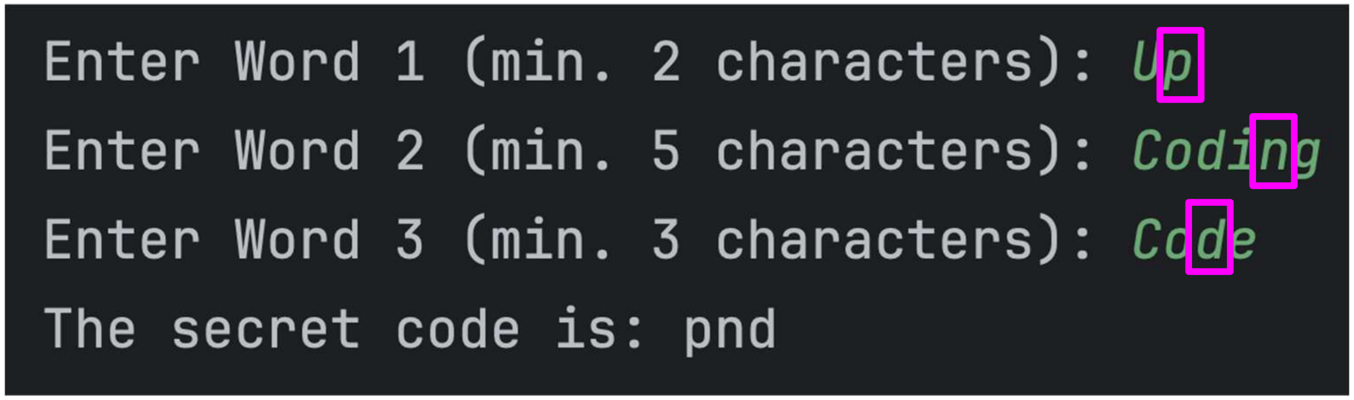
If we want to print “`p`”, why will `print(str[6])` result in an error?

Indexing Activity

Write a program that takes in three words as input. Then, create a secret code using the following format:

2nd letter of str1 + 5th letter of str2 + 3rd letter of str3

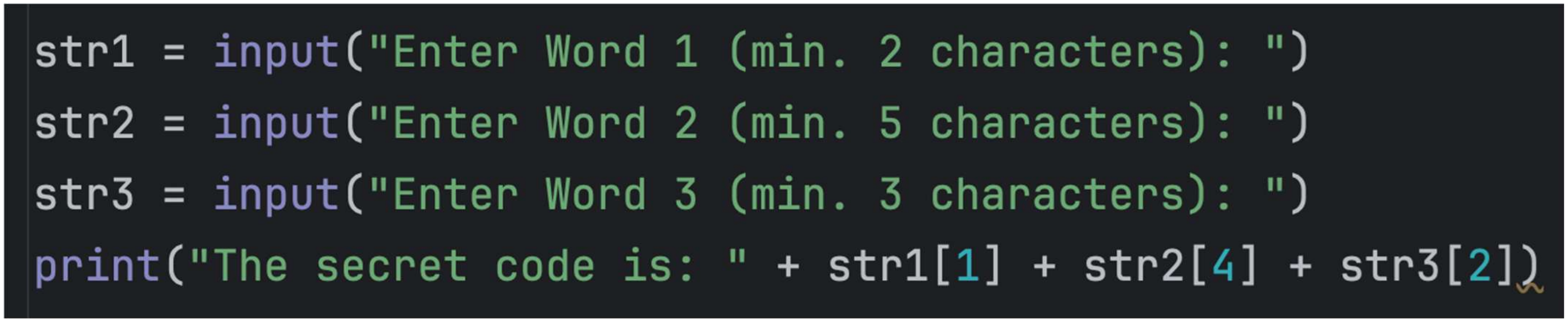
Please ensure that your inputs meet the minimum length requirement, otherwise there will be an error!



```
Enter Word 1 (min. 2 characters): Up
Enter Word 2 (min. 5 characters): Coding
Enter Word 3 (min. 3 characters): Code
The secret code is: pnd
```

Solution

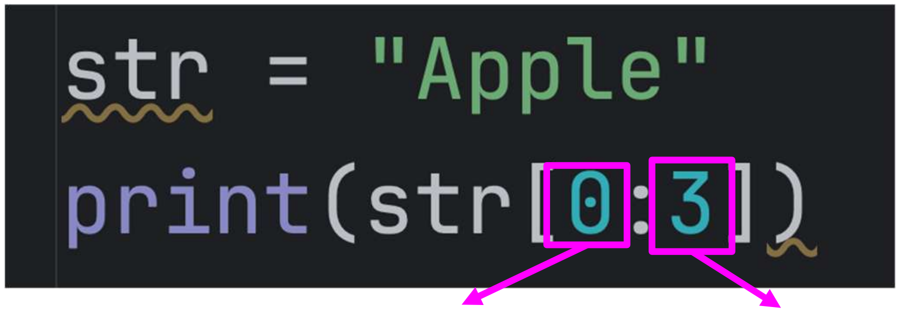
Remember to subtract 1 from the character's position when indexing (ie. 2nd character becomes index 1, 5th character becomes index 4, 3rd character becomes index 2)



```python
str1 = input("Enter Word 1 (min. 2 characters): ")
str2 = input("Enter Word 2 (min. 5 characters): ")
str3 = input("Enter Word 3 (min. 3 characters): ")
print("The secret code is: " + str1[1] + str2[4] + str3[2])
```

Range Indexing

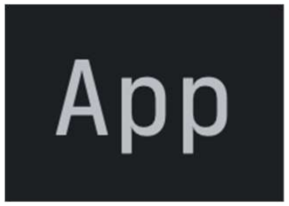
- Python also supports range indexing, where you can extract a substring (smaller string) from the string



```python
str = "Apple"
print(str[0:3])
```

Note: if you put indices that are out of range, it will simply print nothing (the code will **not** return an error)

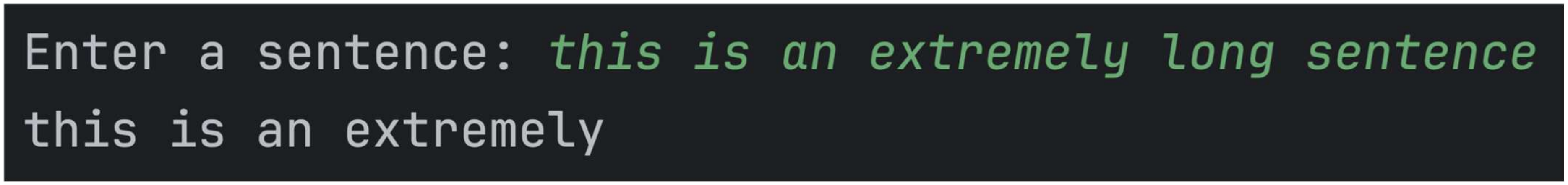
Result:



```
App
```

Range Indexing Activity

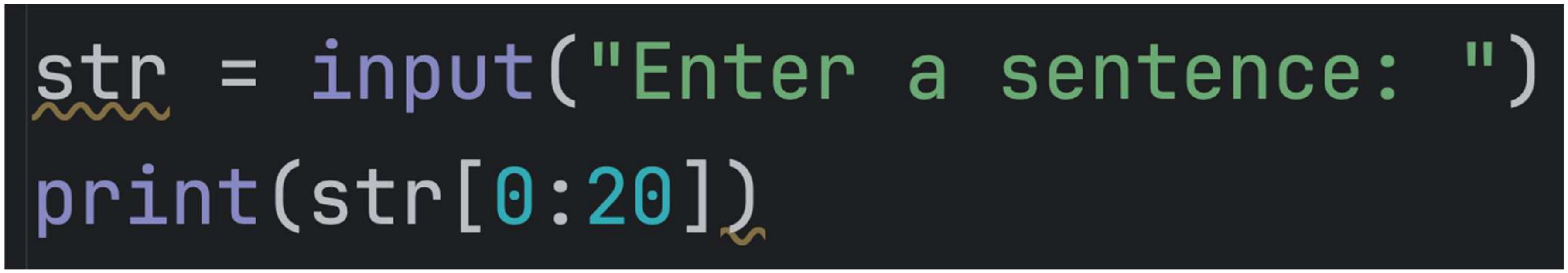
Write a program that takes in a line of input. This can be as long as you would like it to be. Then print the line, but only print the first 20 characters.



```
Enter a sentence: this is an extremely long sentence
this is an extremely
```

Solution

You can simply do `[0:substring_length]`



```python
str = input("Enter a sentence: ")
print(str[0:20])
```

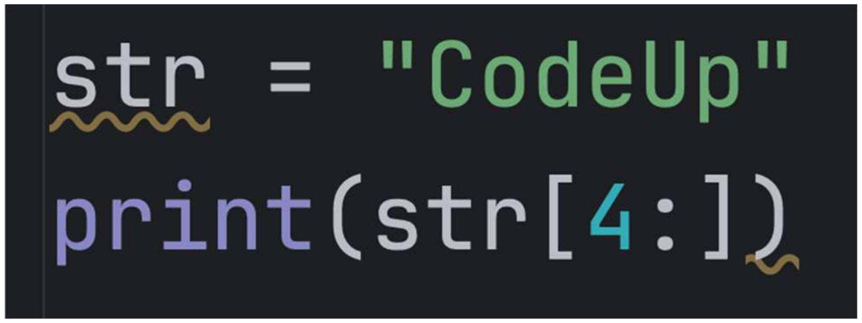
String Splicing

- `str[:i]` - prints everything up to index i (exclusive)
  - The program assumes that the number to the left of `:` is `0` if not specified
- `str[i:]` - prints everything starting from index i (inclusive) to the end of the string
  - The program assumes that the number to the right of `:` is `N - 1` if not specified
- `str[-i]` - prints the i-th last character of the string
- `str[::-1]` - prints the reverse of the string

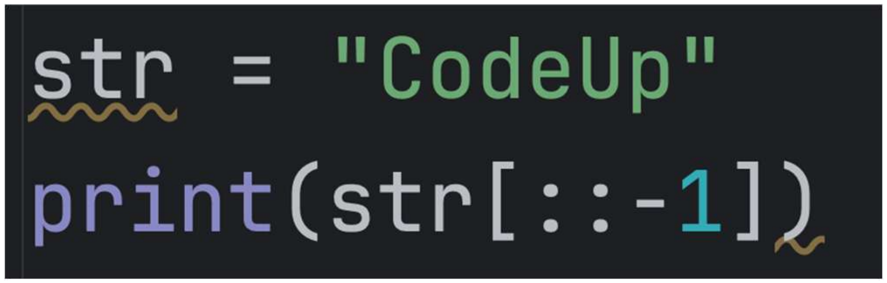
Note: a general rule in Python is the first number of a range is **inclusive** and the second is **exclusive**

Practice

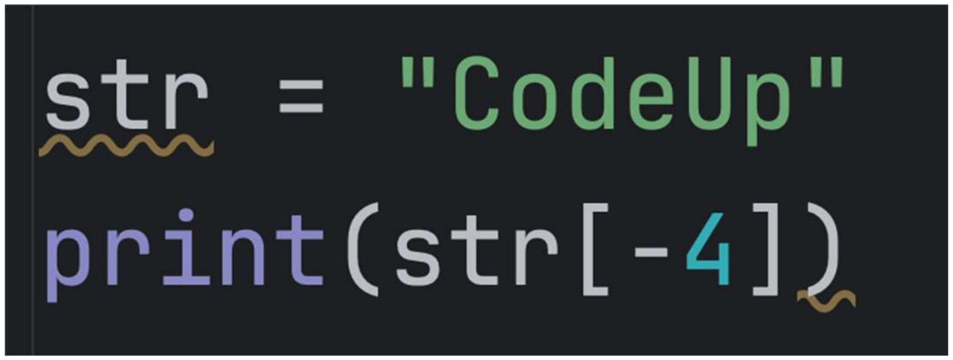
What will the following programs print?



```python
str = "CodeUp"
print(str[4:])
```



```python
str = "CodeUp"
print(str[::-1])
```

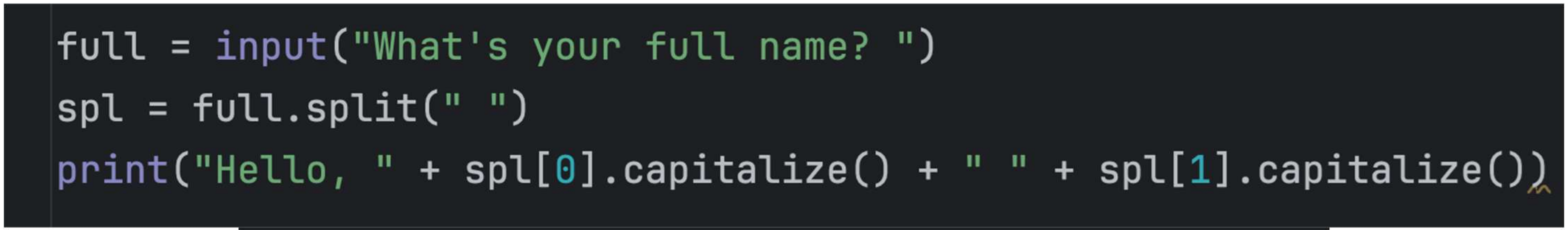


```python
str = "CodeUp"
print(str[-4])
```

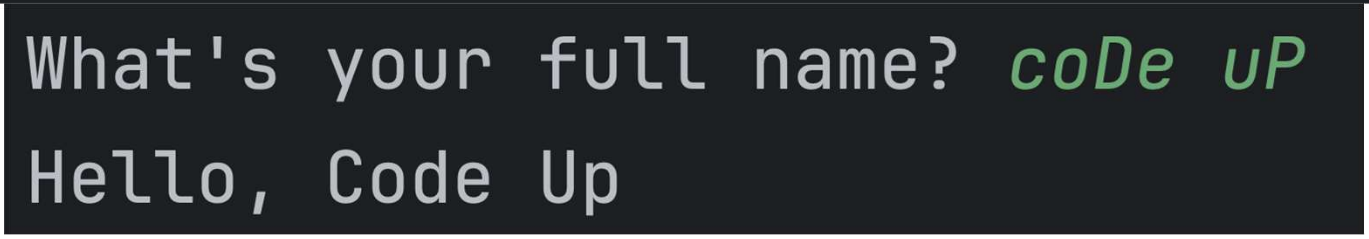
Guided Activity: Full Name Capitalizer

This program allows you to enter your full name (first and last), and will capitalize them both.

The **`str.split()`** function uses lists, we will learn about them later! For now, just know that the function **`split()`** splits strings at spaces.



```
full = input("What's your full name? ")
spl = full.split(" ")
print("Hello, " + spl[0].capitalize() + " " + spl[1].capitalize())
```



```
What's your full name? coDe uP
Hello, Code Up
```

F-Strings

F-strings are a convenience feature, allowing you to *seamlessly* incorporate variables in strings.

ex
Instead of:

```
name = input()
string = “Hello ” + name + “!”
```

You could do:

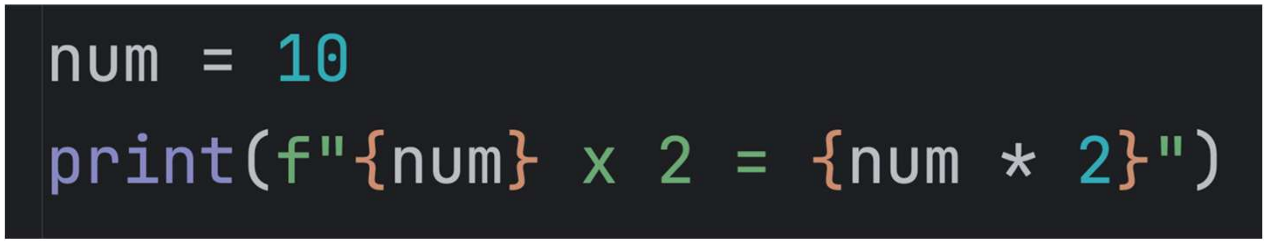
```
name = input()
string = f“Hello {name}!”
```

If you were to input `John`, both would give the same result: `“Hello John!”`

F-strings also avoids the need to typecast non-strings into strings

F-Strings

- Alternatively, you can directly format strings in a print statement instead of storing it in a variable
- When you want to create an f-string, put an “f” before you use quotes and any variables enclosed in curly-brace brackets: {}
    - `money = 10`
    - `print(f"I have {money} dollars")`



```
num = 10
print(f"{num} x 2 = {num * 2}")
```

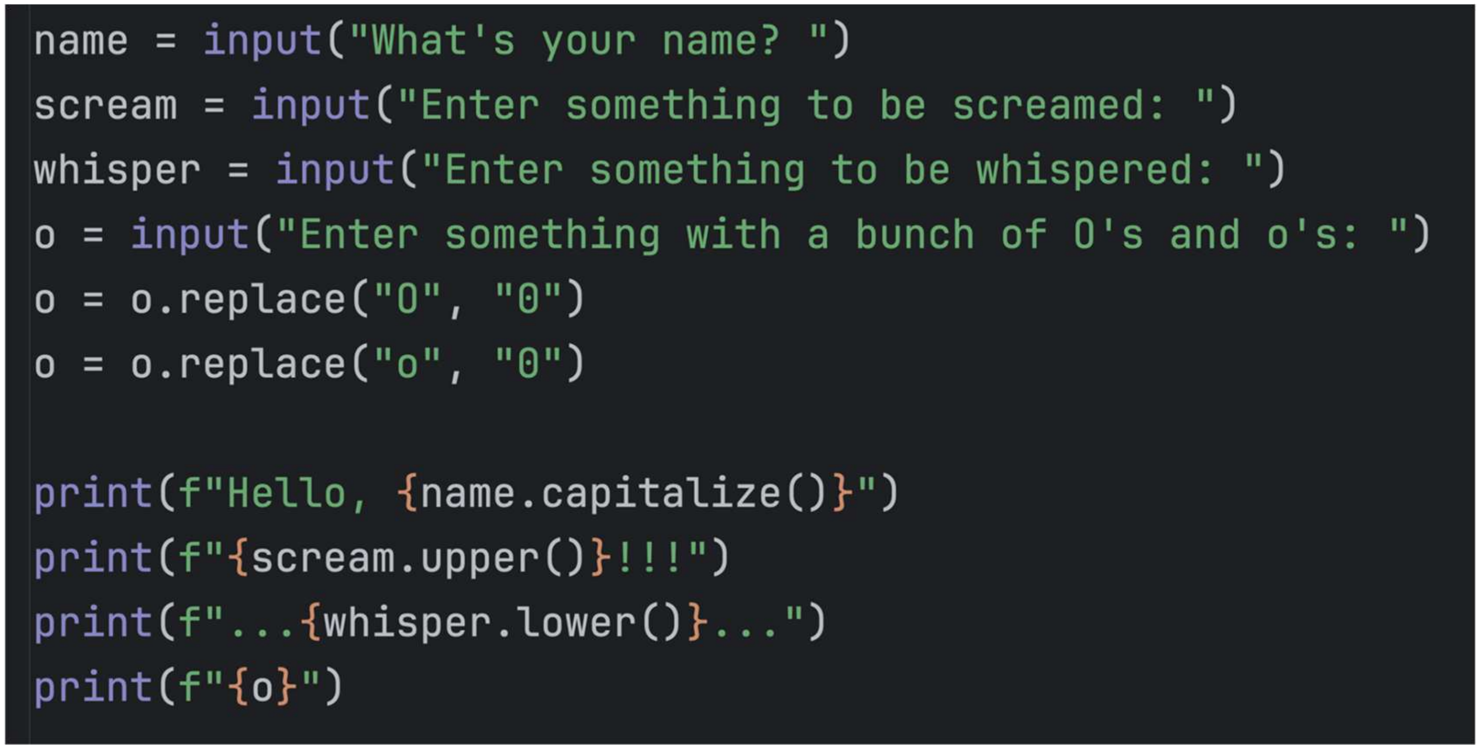
- What will this print out?

F-Strings Activity

Modify your code from the previous activities to use f-string formatting instead of string concatenation

- String Functions Activity
- Indexing Activity
- Range Indexing Activity

Example Solution: String Functions Activity



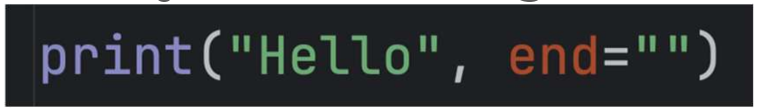
```python
name = input("What's your name? ")
scream = input("Enter something to be screamed: ")
whisper = input("Enter something to be whispered: ")
o = input("Enter something with a bunch of O's and o's: ")
o = o.replace("O", "0")
o = o.replace("o", "0")

print(f"Hello, {name.capitalize()}")
print(f"{scream.upper()}!!!")
print(f"...{whisper.lower()}...")
print(f"{o}")
```

Print end=

The print statement by default adds a newline (creates a new line) to the end of the statement printed. Therefore, the ending by default is “`\n`” (newline)

However, you can change the ending by using the end parameter.

Ex. 

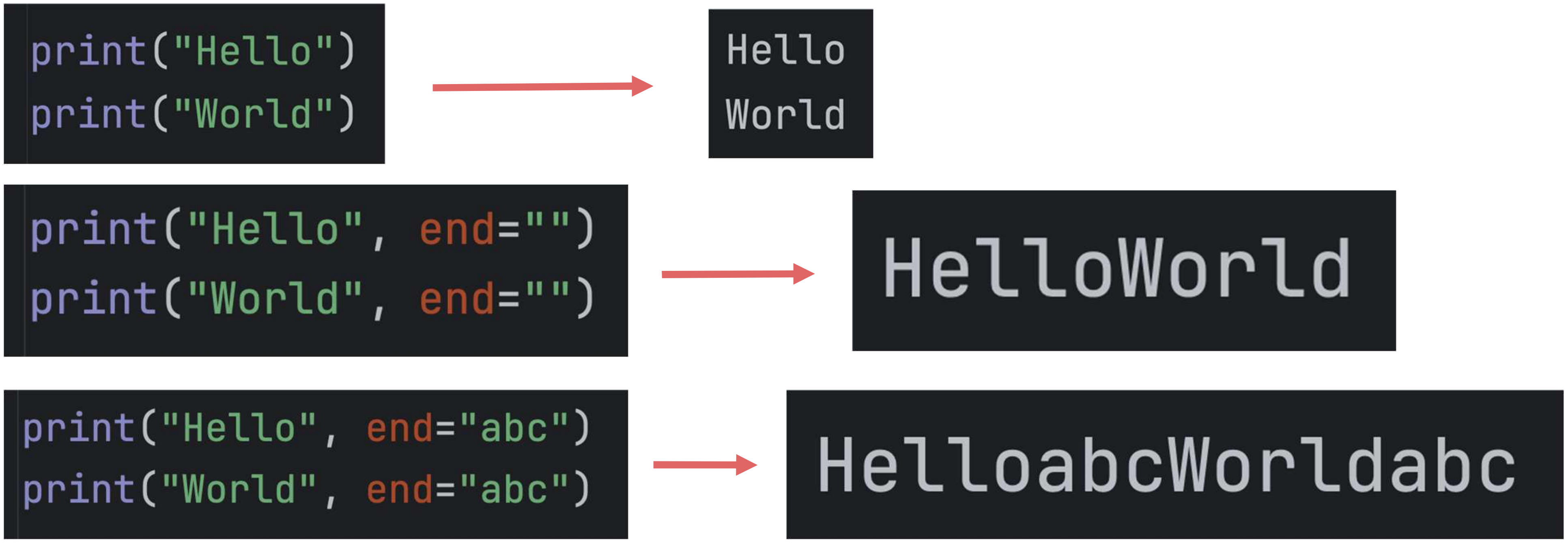
```
print("Hello", end="")
```

- Prints the same statement but without the newline because the ending is set to “” → equivalent to nothing

This way no newline is created → the next output from the code will be written on the same line (in most cases).

Print end=

Here is a side by side comparison of three `print()` statements with different `end=` statements



```
print("Hello")
print("World")
```

```
Hello
World
```

```
print("Hello", end="")
print("World", end="")
```

```
HelloWorld
```

```
print("Hello", end="abc")
print("World", end="abc")
```

```
HelloabcWorldabc
```

Print end= Activity

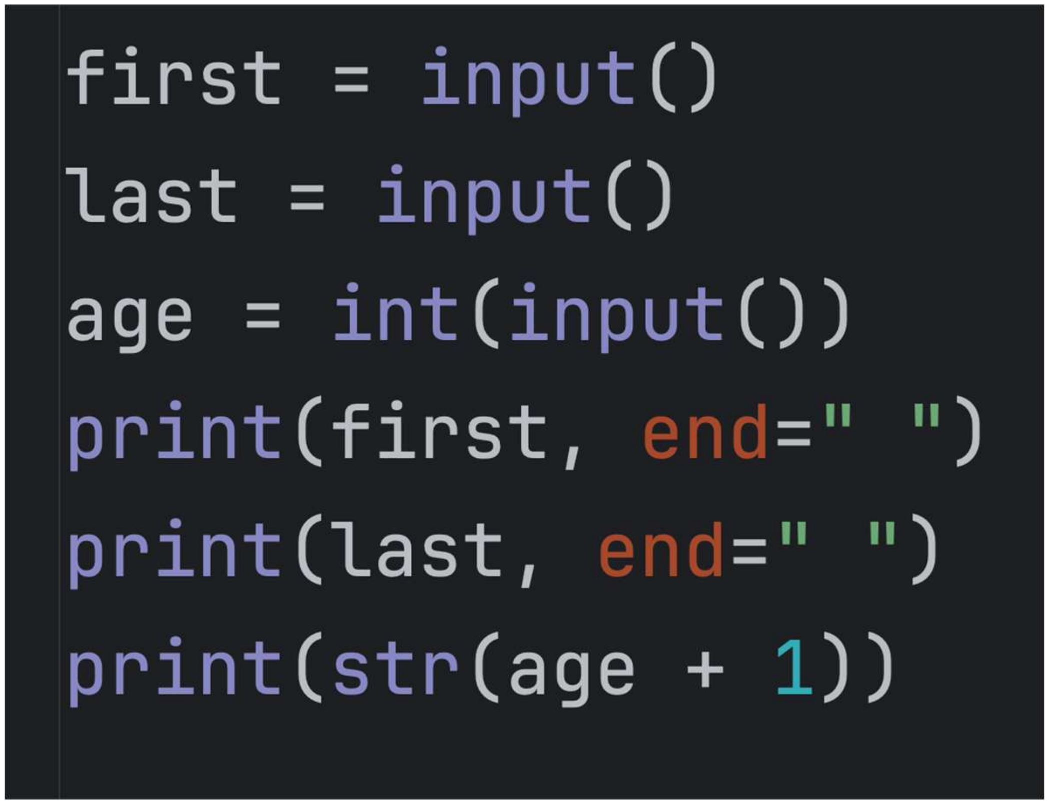
Write a program that takes in somebody's *first name, last name, and age* ***next year***, then prints them both on one line **without** string concatenation or f-strings

Hint: you will need three `print()` statements

i.e.

Input:	Output:
John	John Doe 28
Doe	
27	

Solution

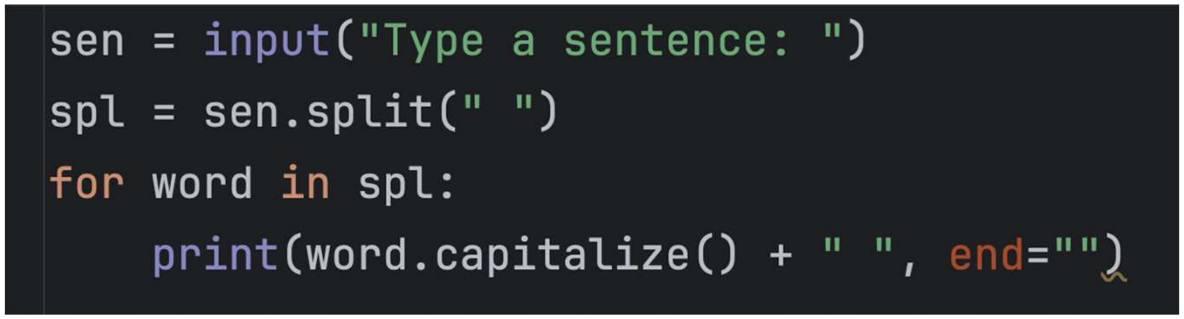


```python
first = input()
last = input()
age = int(input())
print(first, end=" ")
print(last, end=" ")
print(str(age + 1))
```

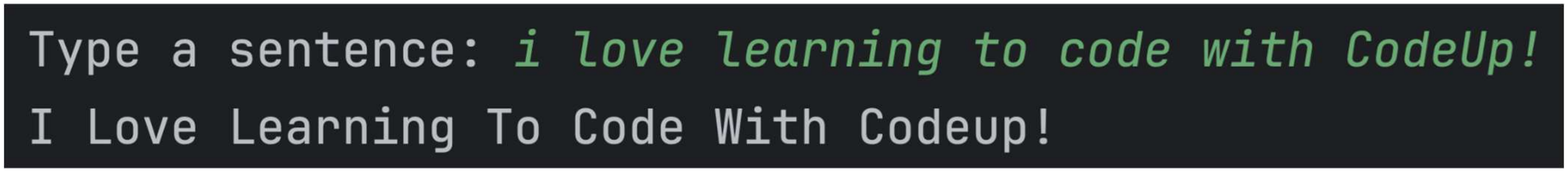
Sentence Capitalizer

This program allows you to enter an entire sentence, and capitalizes only the first letter of each word.

We can use a for loop to loop the entire list (this contains all the individual words split by “ ”). We will learn more about loops next class.



```python
sen = input("Type a sentence: ")
spl = sen.split(" ")
for word in spl:
    print(word.capitalize() + " ", end="")
```

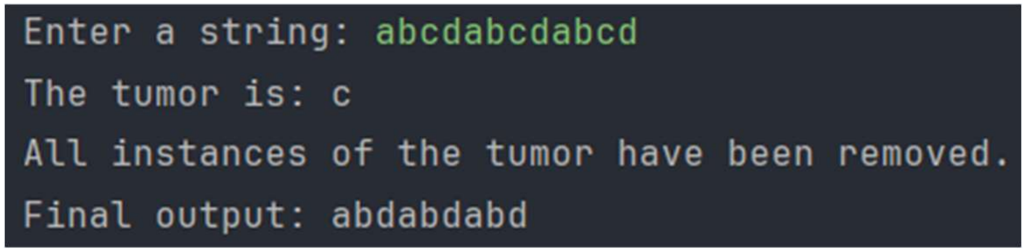


```
Type a sentence: i love learning to code with CodeUp!
I Love Learning To Code With Codeup!
```

Homework

Write a program that will:

- Ask the user to input a string
- Select a random character in the string - call this the ‘tumor’
- Tell the user what the tumor is, and remove all instances of the tumor in the string
- Print out the resulting word after the character has been removed
- Note: Make sure you don’t select an index outside the string’s bounds



```
Enter a string: abcdabcdabcd
The tumor is: c
All instances of the tumor have been removed.
Final output: abdabdabd
```