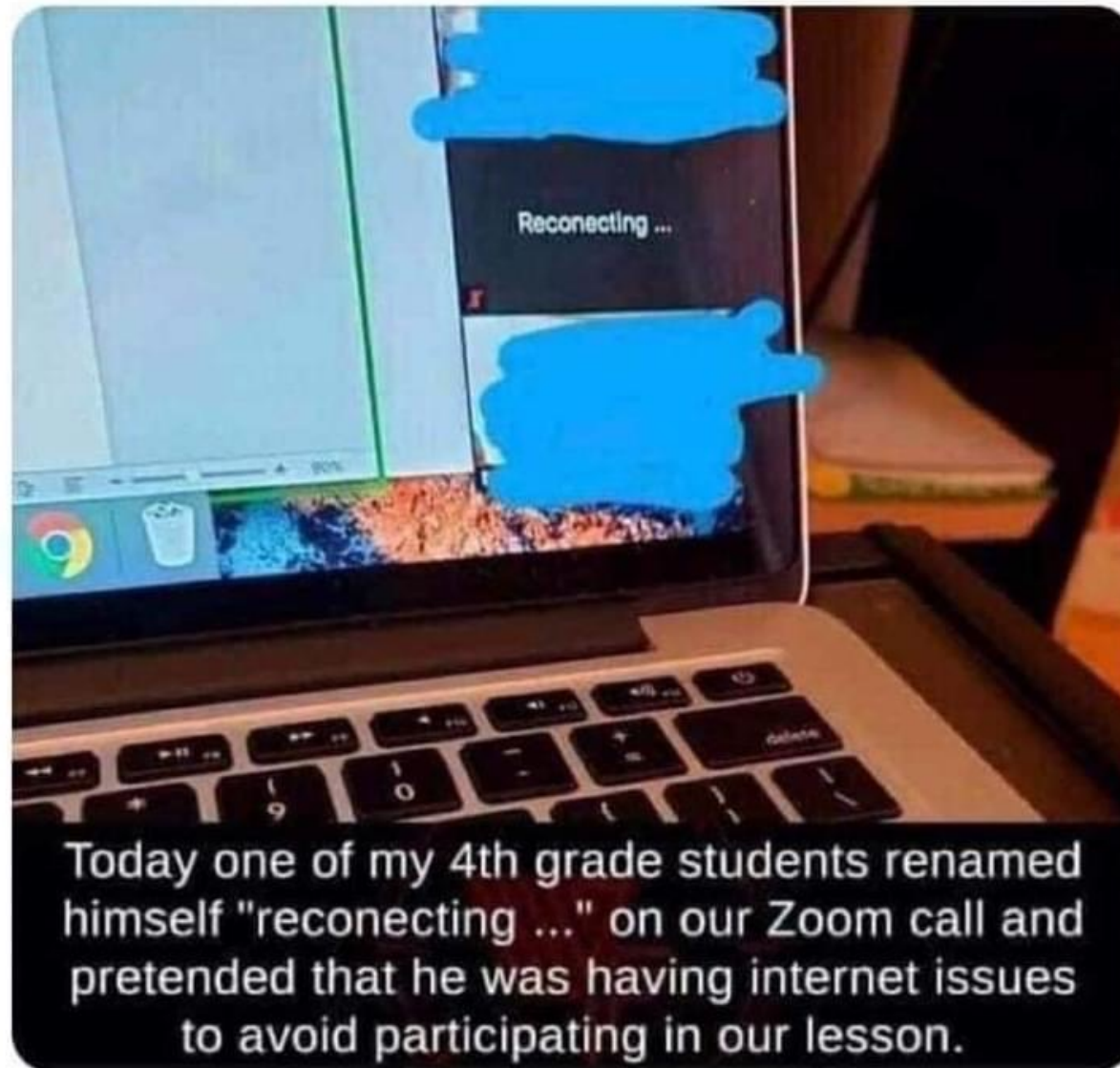


Introduction to Problem Solving in Python

COSI 10A

The future of IT is in good hands.



Zoom
Classes



Class objectives

- More on Strings
- Conditional Execution & Return



Review: Evaluating logical expressions

- Relational operators have lower precedence than math; logical operators have lower precedence than relational operators

```
5 * 7 >= 3 + 5 * (7 - 1) and 7 <= 11
```

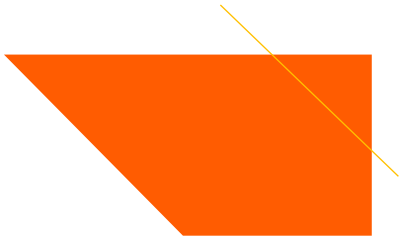
```
5 * 7 >= 3 + 5 * 6 and 7 <= 11
```

```
35 >= 3 + 30 and 7 <= 11
```

```
35 >= 33 and 7 <= 11
```

```
True and True
```

```
True
```



Strings



Modifying strings

- String operations and functions like `lowercase` build and return a new string, rather than modifying the current string

```
s = "Test"  
s.upper()  
print(s)    # Test
```

Strings are immutable. The value cannot change

- To modify a variable's value, you must reassign it:

```
s = "Test"  
s = s.upper()  
print(s)    # TEST
```



Looping through a string

- The `for` loop through a string using **`range`**:

```
major = "CS"  
for letter in range(0, len(major)):  
    print(major[letter])
```

- You can also use a `for` loop to print or examine each character without `range`

```
major = "CS"  
for letter in major:  
    print(letter)
```



String tests

Method	Description
<code>startswith(str)</code>	whether one contains other's characters at start
<code>endswith(str)</code>	whether one contains other's characters at end

```
name = "Anastasia"  
if name.startswith("Anas"):  
    print("check 1")
```

🟡 The `in` keyword can be used to test if a string contains another string.

```
"sta" in name      # true
```




String question 1

- Write a function called `longest_name` that accepts an integer `n` as a parameter and prompts for `n` names, then prints the longest name. For example: `longest_name(4)`

```
name 1? Roy
name 2? Dane
name 3? Marina
name 4? Hercules
Hercules is the longest name
```



String question 1

- Write a function called `longest_name` that accepts an integer `n` as a parameter and prompts for `n` names, then prints the longest name. For example: `longest_name(4)`

```
name 1? Roy
name 2? Dane
name 3? Marina
name 4? Hercules
Hercules is the longest name
```

```
def longest_name(n):
    longest = input("name " + str(1) + "? ")
    for i in range(2, n+1):
        name = input("name " + str(i) + "? ")
        if len(name) > len(longest):
            longest = name
    print(longest, "is the longest name")
```



Strings and ints

- Individual characters in a string are stored internally as integers
- A standard encoding (*ASCII* value) determines which integer value represent each character

Examples:

'A' is 65, 'B' is 66, ' ' is 32
'a' is 97, 'b' is 98, '*' is 42

- One character long `Strings` and `ints` can be converted to each other

`ord('a')` is 97, `chr(103)` is 'g'

- This is useful because you can do the following:

`chr(ord('a') + 2)` is 'c'



String question 2

What output is produced by the following program?

```
def print_range(start_letter, end_letter):  
    for i in range (ord(end_letter) - ord(start_letter) + 1):  
        letter = chr(ord(start_letter) + i)  
        print(letter, end="")  
    print()  
  
def main():  
    print_range("a", "z")  
    print_range("e", "g")  
    print_range("z", "a")  
  
main()
```



String question 3

- Write an if statement that tests to see whether a string begins with an uppercase letter



String question 3

Write an if statement that tests to see whether a string begins with an uppercase letter

1. `if "A" <= the_string[0] <= "Z":`

2. `if the_string[0] >= "A" and the_string[0] <= "Z":`



String question 4

- Write a function that calculates the length of a string.



String question 4

- Write a function that calculates the length of a string.

```
def string_length(str1):  
    count = 0  
    for char in str1:  
        count += 1  
    return count
```




Returning from `if`



When to return?

- Functions with loops and return values can be tricky. When and where should the function return its result?
- Write a function `seven` that asks the user for ten numbers from 1-30. If any of the numbers is a lucky 7, the function should stop and return `True`. If none of the ten are 7 it should return `False`.

```
Please enter a number from 1 to 30: 4
Please enter a number from 1 to 30: 6
Please enter a number from 1 to 30: 3
Please enter a number from 1 to 30: 2
Please enter a number from 1 to 30: 7
```



Flawed solution

```
def seven():  
    for i in range(10):  
        num = int(input("Please enter a number from 1 to 30: "))  
        if num == 7:  
            return True  
        else:  
            return False  
  
def main():  
    print(seven())  
  
main()
```

- ❖ The function always returns immediately after the first number
- ❖ This is wrong if that number isn't a 7



Returning at the right time

```
def seven():  
    for i in range(10):  
        num = int(input("Please enter a number from 1 to 30: "))  
        if num == 7:  
            return True  
    return False  
  
def main():  
    print(seven())  
  
main()
```

- ❖ Returns `True` immediately if 7 is found
- ❖ If 7 isn't found, the loop continues
- ❖ If all ten aren't 7, the loop ends and we return `False`



if/else return question

- Write a function `count_factors` that returns the number of factors of an integer
- `count_factors(24)` returns 8 (the factors of 24 are 1, 2, 3, 4, 6, 8, 12, and 24)