CSE 110

Home W1 W2 W3 W4 W5 W6 W7

Learning Activity (2 of 2): Data in Files

Prepare

Overview

In this lesson, you will continue applying all the topics of this semester as you work with data in files.

Preparation Material

For the most part, you know all the topics you need to complete this lesson. But, there is one additional strategy that may be helpful.

Finding the Max or Min

Recall from previous lessons that a way to find the largest number in a list is to keep track of the largest number found to date, and then update that value if you find a new one that is larger. For example:

```
numbers = [42, 25, 18, 83, 23, 85, 38, 2]

largest_so_far = numbers[0]

for number in numbers:
    if number > largest_so_far:
        # This number is larger than the largest we had seen so far

        # So it is now the largest we've seen
        largest_so_far = number

# Now, after the loop we can display it:
print(f"The largest is: {largest_so_far}")
```

This produces the following output:

```
The largest is: 85
```

It is important to remember to start the **`largest_so_far`** variable at either the first value (assuming it's not an empty list) or else something that you know will be smaller than everything in your list. Otherwise, you won't consider some numbers. For example, if you started it at 10, then you would never consider anything less than 10.

Keeping track of the item that is the Max or Min

What if you want to know not only which number is the largest, but also the item that is the largest? For example, you may want to know not only the most expensive price in your shopping cart, but the actual product that is the most expensive. The logic above will only find the largest value. To find the largest item, you'll also need to keep track of its name or value along the way.

Consider the following list of products in a shopping cart, along with their prices:

```
shopping_cart = [
    ["Chips", 2.99],
    ["Bread", 2.50],
    ["Milk", 3.19],
    ["Ice Cream", 6.99],
    ["Cheese", 5.99],
    ["Candy bar", 0.99]
]
```

In order to find the most expensive price, we can iterate through it as before:

```
max_price = -1

for item in shopping_cart:
    price = item[1] # The price is the second part of the item

    if price > max_price:
        # This is the new max
        max_price = price

print(f"The maximum price is: {max_price}")
```

This produces the following output:

```
The maximum price is: 6.99
```

If we want to find the name of the item that is the most expensive, we need to declare a variable (for example, **`max_product`**) before the loop to keep track of the *product* that had the maximum price. Then, whenever we update the max_price to save a new price, we also update the max_product to be the corresponding product name. For example:

```
max_price = -1
max_product = "" # It doesn't matter what you set this to, it just need

for item in shopping_cart:
    product_name = item[0] # Product name is the first part
    price = item[1]

    if price > max_price:
        # This is the new max
        max_price = price

        # Also save this product name as the max product
        max_product = product_name

print(f"The maximum price is: {max_price}")
print(f"The product with the maximum price is: {max_product}")
```

This produces the following output:

```
The maximum price is: 6.99
The product with the maximum price is: Ice Cream
```

Other Code with Loops and Files

In addition to the code samples you've seen here, keep in mind that you can use all of the other components you have learned throughout the semester in conjunction with files. For example, if you wanted to restrict your analysis of products to only those that had a price over a certain amount, or that had a name matching certain criteria, you can include an if statement in the middle of the loop that is iterating through the file.

In many ways, this lesson is the culmination of all the building blocks that you've learned this semester and brings them all together.

In this activity, you will continue practicing working with `if` statements, using even more complicated programs and logic.

Activity Instructions

Overview

Practice finding elements in a list.

Instructions

To simplify your program to focus on the task at hand, instead of reading from a file, please copy and paste the following list of people and their ages into your program:

```
people = [
    "Stephanie 36",
    "John 29",
    "Emily 24",
    "Gretchen 54",
    "Noah 12",
    "Penelope 32",
    "Michael 2",
    "Jacob 10"
]
```

Write a program to find the youngest person in the list.

Consider following these steps to build up to the finished program:

1. Iterate through the list and display each string to the screen.
2. Split the string into a name and age and print them to the screen.
3. Find the age that is the youngest.
4. Keep track of the name of the person that is the youngest.

Sample Solution

When your program is finished, please view a sample solution of this program to compare your approach to that one.

You should work to complete this checkpoint program first, without looking at the sample solution. However, if you have worked on it for at least an hour and are still having problems, you may feel free to use the sample solution to help you finish your program.

- Sample solution

Testing Procedure

Because there is no user input, you don't need to test the program with different values. Instead, make sure that that following components are in place:

1. The list of people is included in the program.
2. The program can iterate through each string in the list.
3. The program can split the string into the appropriate pieces.
4. The youngest age is identified.
5. The name of the youngest person is identified.

Submission

You have now completed all of the learning activities for the week!

Make sure to:

- Return to Canvas and submit the associated quiz.

Up Next

- Check Your Understanding

Other Links:

- Return to: Week Overview | Course Home