Day 15

Problem Decomposition
Announcements

● Lab 2 AutoLab will be open for submissions by tonight — Please make sure to submit sooner rather than later
Recap

...we've covered a lot of stuff so far
Recap

...we've covered a lot of stuff so far

*How do we know when to use it?*
How do we store values in our programs?

- We can use variables to store a single value
- We can use ordered collections (lists/arrays) to store multiple values
- We can use associative collections (dictionaries/objects) to store multiple key-value pairs

Why do we store values in our programs?

- So we can use them later (like making a note or reminder for yourself)
- To give a value a name/meaning (ie: \( \pi = 3.14159 \))
Storing Values

How do we store values in our programs?
- We can use *variables* to store a single value
- We can use *ordered collections* (lists/arrays) to store multiple values
- We can use *associative collections* (dictionaries/objects) to store multiple key-value pairs

Why do we store values in our programs?
- So we can use them later (like making a note or reminder for yourself)
- To give a value a name/meaning (ie: pi = 3.14159)
How would we store the following?

- Someone's name
- A grocery list
- The name and price of an item
- A stock market quote
- A receipt
- The population of a country
- Words in a book
- A student's name, major, and year
- A class roster with names and grades
How would we store the following?

**Someone's name**
- A grocery list
- The name and price of an item
- A stock market quote
- A receipt

**Variables**

**The population of a country**
- Words in a book
- A students name, major, and year
- A class roster with names and grades
Storing Values

How would we store the following?

- Someone's name
- A grocery list
- The name and price of an item
- A stock market quote
- A receipt
- The population of a country
- Lists
- Words in a book
- A student's name, major, and year
- A class roster with names and grades
How would we store the following?

Someone's name  
A grocery list  
The name and price of an item  
A stock market quote  
A receipt  
The population of a country  
Words in a book  
A students name, major, and year  
A class roster with names and grades
How would we store the following?

- Someone's name
- A grocery list
- The name and price of an item
- A stock market quote
- A receipt
- The population of a country
- Words in a book
- A students name, major, and year
- A class roster with names and grades
- A list of dictionaries
Defining Tasks

A function allows you to define a task.

Functions have inputs and an output:
1. Do something with the inputs
2. Potentially have other effects, i.e., printing something
3. Produce an output
Defining Tasks

1. eat("Cereal")
2. driveTo("Work")
3. work(["CSE487", "CSE503"])
4. eat("Pizza")
5. work(["Make slides", "TA meeting", "CSE250"])
6. driveTo("Home")
7. eat("Spaghetti")
Defining Tasks

1. eat("Cereal")
2. driveTo("Work")
3. work(["CSE487", "CSE503"])
4. eat("Pizza")
5. work(["Make slides", "TA meeting", "CSE250"])
6. driveTo("Home")
7. eat("Spaghetti")

The same function can be called with different inputs
Making Decisions

If statements are used to make decisions...

- You can choose to do something conditionally:
  - I will only wear a jacket if it is cold
  - If it is a weekday, I will go to work

- You can choose between multiple options:
  - I will order strawberry if they have it, otherwise I will order vanilla
  - If you have above a 90, you will get an A. If instead you have above and 80, you will get a B...
Making Decisions

If statements are used to make decisions...
You can choose to do something conditionally:

- I will only wear a jacket if it is cold
- If it is a weekday, I will go to work
Making Decisions

If statements are used to make decisions...
You can choose to do something conditionally:
- I will only wear a jacket if it is cold
- If it is a weekday, I will go to work

You can choose between multiple options:
- I will order strawberry if they have it, otherwise I will order vanilla
- If you have above a 90, you will get an A. If instead you have above and 80, you will get a B...
Making Decisions

If statements are used to make decisions...

You can choose to do something conditionally:

- I will only wear a jacket if it is cold
- If it is a weekday, I will go to work

You can choose between multiple options:

- I will order strawberry if they have it, otherwise I will order vanilla
- If you have above a 90, you will get an A. If instead you have above and 80, you will get a B...

Language to look for:

- if, check, when, instead, otherwise, choose, select, which
Repeating Tasks

To repeat a task multiple times, we use a loop...
Repeating Tasks

To repeat a task multiple times, we use a loop...

Often used with collections (ordered and associative)

- Do something with every item in a collection
- Search for something specific in a collection
- Accumulate some value (sum, product, count, etc)
Repeating Tasks

To repeat a task multiple times, we use a loop...

Often used with collections (ordered and associative)
- Do something with every item in a collection
- Search for something specific in a collection
- Accumulate some value (sum, product, count, etc)

Language to look for:
- all, for each, each, every, times, find, total
Repeating Tasks

- Print "hello" 10 times
- Calculate the total price of every item in your shopping cart
- Email every student in class
- Put away all your books
- Find the longest book on the shelf
- Check the expiration date of each item in the fridge
A shopping cart dictionary pairs customer names with lists of items they plan to buy. For example:

```
shoppingCarts = {
    'joe': ['milk', 'cookies', 'spinach'],
    'amy': ['carrots', 'flour', 'sugar', 'milk', 'cereal']
}
```

A price list dictionary pairs product names with the prices. For example:

```
priceList = {
    'milk': 1.49, 'cookies': 2.00, 'spinach': 0.49, 'carrots': 1.00,
    'flour': 2.49, 'sugar': 2.29, 'cereal': 1.79
}
```
Define a function named `cartTotals` that takes a shopping cart dictionary and a price list dictionary, and returns a new dictionary of customer names and the total amount they owe for their purchases.
Define a function named `cartTotals` that takes a shopping cart dictionary and a price list dictionary, and returns a new dictionary of customer names and the total amount they owe for their purchases.

...there's a lot going on...
Define a function named `cartTotals` that takes a shopping cart dictionary and a price list dictionary, and returns a new dictionary of customer names and the total amount they owe for their purchases.

...there's a lot going on...

...this is bigger than problems we've solved before...
Big Exercise

Define a function named `cartTotals` that takes a shopping cart dictionary and a price list dictionary, and returns a new dictionary of customer names and the total amount they owe for their purchases.

...there's a lot going on...

...this is bigger than problems we've solved before...

Where do we even begin!?
Problem Decomposition

Take a deep breath...
Problem Decomposition

Take a deep breath...

We have all of the knowledge we need.
Problem Decomposition

Take a deep breath...

We have all of the knowledge we need.

To approach bigger problems, we just need to break them down into smaller sub-problems.
Problem Decomposition

Take a deep breath...

We have all of the knowledge we need.

To approach bigger problems, we just need to break them down into smaller sub-problems.

*What are possible sub-problems for this exercise?*
Problem Decomposition

What are possible sub-problems for this exercise?

Define a function named `cartTotals` that takes a shopping cart dictionary and a price list dictionary, and returns a new dictionary of customer names and the total amount they owe for their purchases.
What are possible sub-problems for this exercise?

For each customer, we have to compute their total cost.
What are possible sub-problems for this exercise?

For each customer, we have to compute their total cost. To do that we have to be able to compute the total cost for one customer.
What are possible sub-problems for this exercise?

For each customer, we have to compute their total cost.
To do that we have to be able to compute the total cost for one customer.
To do that we have to be able to compute the total cost of a cart.
Problem Decomposition

What are possible sub-problems for this exercise?

For each customer, we have to compute their total cost. To do that we have to be able to compute the total cost for one customer. To do that we have to be able to compute the total cost of a cart.

Start with the simplest problem.
Define a function named `cartTotal` that takes a cart (a list of items), and a price dictionary (a dictionary mapping item name to price), and compute the total cost of that cart.
Sub-Problem #1

Define a function named `cartTotal` that takes a cart (a list of items), and a price dictionary (a dictionary mapping item name to price), and compute the total cost of that cart.

```
1. def cartTotal(cart, priceDict):
2.     total = 0
3.     # for each item in the cart...
4.     # add its price to the total...
5.     return total
```
Sub-Problem #1

Define a function named `cartTotal` that takes a cart (a list of items), and a price dictionary (a dictionary mapping item name to price), and compute the total cost of that cart.

1. `def cartTotal(cart, priceDict):`
2. `total = 0`
3. `for item in cart: # for each item in the cart...`
4. `# add its price to the total...`
5. `return total`
Define a function named `cartTotal` that takes a cart (a list of items), and a price dictionary (a dictionary mapping item name to price), and compute the total cost of that cart.

1. `def cartTotal(cart, priceDict):
2.     total = 0
3.     for item in cart: # for each item in the cart...
4.         total = total + priceDict[item] # add its price to the total...
5.     return total
Define a function named `customerCartTotal` that takes a customer name, a shopping cart dictionary, and a price list dictionary. The function should return the total cost of the customers cart.
Define a function named `customerCartTotal` that takes a customer name, a shopping cart dictionary, and a price list dictionary. The function should return the total cost of the customer's cart.

```python
def customerTotal(customer, carts, prices):
    # Get the customer's cart...
    # Compute the total cost of the cart
```
Define a function named `customerCartTotal` that takes a customer name, a shopping cart dictionary, and a price list dictionary. The function should return the total cost of the customer's cart.

```python
def customerTotal(customer, carts, prices):
    cart = carts[customer]  # Get the customer's cart...
    # Compute the total cost of the cart
```
Sub-Problem #2

Define a function named `customerCartTotal` that takes a customer name, a shopping cart dictionary, and a price list dictionary. The function should return the total cost of the customers cart.

```python
def customerTotal(customer, carts, prices):
    cart = carts[customer]  # Get the customer's cart...
    # Compute the total cost of the cart
```

We just solved this problem!!
Sub-Problem #2

Define a function named \texttt{customerCartTotal} that takes a customer name, a shopping cart dictionary, and a price list dictionary. The function should return the total cost of the customers cart.

\begin{verbatim}
def customerTotal(customer, carts, prices):
    cart = carts[customer]  # Get the customer's cart...
    return cartTotal(cart, prices)  # Compute the total cost of the cart
\end{verbatim}
The Original Problem

Define a function named `cartTotals` that takes a shopping cart dictionary and a price list dictionary, and returns a new dictionary of customer names and the total amount they owe for their purchases.
The Original Problem

Define a function named `cartTotals` that takes a shopping cart dictionary and a price list dictionary, and returns a new dictionary of customer names and the total amount they owe for their purchases.

1. `def cartTotals(carts, prices):
2.     result = {}
3.     # For each customer...
4.     # Compute their total cost and add it to the result
5.     return result`
Define a function named `cartTotals` that takes a shopping cart dictionary and a price list dictionary, and returns a new dictionary of customer names and the total amount they owe for their purchases.

1. def `cartTotals`(carts, prices):
2.     result = {}
3.     for customer in carts.keys():  # For each customer...
4.         # Compute their total cost and add it to the result
5.     return result

We just solved this problem!!
The Original Problem

Define a function named `cartTotals` that takes a shopping cart dictionary and a price list dictionary, and returns a new dictionary of customer names and the total amount they owe for their purchases.

1. def cartTotals(carts, prices):
2.     result = {}
3.     for customer in carts.keys(): # For each customer...
4.         # Compute their total cost and add it to the result
5.         result[customer] = customerTotal(customer, carts, prices)
6.     return result