Vending Machine Project-

* Contains several products to buy
  + Just need to load a dictionary
* Each product has a name, category, and price
  + Full list is name, category, price, current-stock, max-stock(, stock-status- idea that can be replaced with if statement but also still works)
* Some example categories are drink, crisps, chocolate bars and cereal bars.
  + Basic categories
* The machine can only hold a certain amount of each product.
  + Max-stock
* On start-up a random number of each product is stocked
  + Need to iterate through dictionary and call in .math random number and then set the upper bounds to max-stock
* After each turn the machine should display all products, ordered by category and the name
* If a product is in stock it should display how many are left
  + Whenever you are buying something it should display its name, category, price, current-stock
* If a product is out of stock, it should just display out of stock
  + An if statement is run on user input purchase where it checks if current-stock = 0

Dictionary containing name, category, price, current-stock, max-stock, stock-status, total-value

Machine should accept several commands:

* Buy - allow a user to buy a particular product. They are prompted for money and must enter a valid amount of money. Machine should ‘dispense’ the product and the correct change.
  + User inputs a name using scanner, program searches through dictionary and matches input with item, then displays price, prompts user to pay that price, if matches then given and stock number goes down, if doesn’t match the request is denied and given the option to try again with a y/n user prompt.
* Refill - allows a user to refill a product with an amount. User is prompted for an amount to add. Must not exceed capacity for that product.
  + User inputs that they would like to stock the machine (from several options), Simple adds value to object, an if statement check is run to see if it exceeded max-stock at which point it states overflow and sets it to the max value instead.
* Stock - allows user to take a stock take of the unsold products. Should display each product with quantity unsold and value. Should also print a total value for all stock.
  + Iterates through dictionary, multiplies current-stock by price.

User experience:

Programs loads up and the user receives 4 prompts:

* 1) Buy
* 2) Refill Item
* 3) See What Is Available
* 4) Stock Check

Program calls for a user input and states that they must select a number, if there is an incorrect number it will return “Invalid option, please try again”

1) Upon clicking buy:

* User receives a prompt to enter “What would you like to buy?”
  + If bad entry then states “Invalid entry” and prompts user to try again
  + If item is out of stock then states “Item is out of stock and prompts user to try again
  + If entry matches an item displays its name, category, price, current-stock.
    - Prompt the user with the option “Would you like to buy {item-name}? y/n”
      * If user selects y then another prompt is displayed on the screen “That will be {item-price}, please ”
        + User is then prompted to enter that amount in.

If the amount matches the item price then program states “Thank you for your purchase!” and then the current-stock for the item will go down (iterates through, finds name, reduces stock).

If the amount does not match the item price then program states “Insufficient funds” and returns back to “Would you like to buy {item-name}”.

* + - * If user selects n then returns to “What would you like to buy?”
  + If entry is n or nothing, returns back to menu.

2) Upon clicking refill Item:

* User receives a prompt, what item would you like to refill and by how much?
  + If valid entry then takes name and price, iterates through the inventory to find the object and then increases the stock amount by the set value
    - If stock amount set is above the max stock then program states “Item stock filled to max capacity of {max-stock}”
  + If bad entry then states “Invalid entry” and prompts user to try again

3) See what is available

* Prints the entire catalogue and then ends with “Would you like to return to the menu? y/n”
  + If y is inputted returns to menu
  + If n is inputted nothing happens

4) Stock Check

* Prints the entire catalogue with name, price, current-stock, total-value while producing an end sum of the amount(which is achieved by iterating through and tallying total-value) and then ends with “Would you like to return to the menu? y/n”
  + If y is inputted returns to menu
  + If n is inputted nothing happens