Sure, here are two potential improvements for the code and the steps to implement them:

**Improvement 1: Adding Constants**

* Issue: The product options are hardcoded in the display\_available\_items method. This could be improved by defining them as constants at the beginning of the code.
* Steps:
  1. Create a new branch: git checkout -b add-constants
  2. In the code, move the product definitions to the top and make them constants.
  3. Commit the changes: git commit -m "Moved product definitions to constants"
  4. Push your work: git push origin add-constants
  5. Create a PR for your changes.
  6. Merge the PR and close the issue.

**Improvement 2: Renaming Variables and Functions**

* Issue: Some variable and function names could be more descriptive. For example, cart could be renamed to items\_in\_basket, and add\_to\_cart could be renamed to add\_item\_to\_basket.
* Steps:
  1. Create a new branch: git checkout -b rename-vars-funcs
  2. In the code, rename the variables and functions as described above.
  3. Commit the changes: git commit -m "Renamed variables and functions for clarity"
  4. Push your work: git push origin rename-vars-funcs
  5. Create a PR for your changes.
  6. Merge the PR and close the issue.

Please note that these steps assume you are using Git for version control and that you have a remote repository where you can push your changes and create PRs. If you’re not using Git, the steps for creating branches, committing changes, and creating PRs would be different. Also, remember to test your code after making these changes to ensure everything still works as expected. If you encounter any issues or have any questions, feel free to ask! I’m here to help. 😊

Sure, here’s how your code would look after implementing both improvements:

**After Improvement 1: Adding Constants**

# Define available items using a dictionary

AVAILABLE\_ITEMS = {

"1": PetItem("Dog Chow", 20.99),

"2": PetItem("Kitty Toy", 8.50),

"3": PetItem("Bird House", 35.75),

}

class PetItem:

def \_\_init\_\_(self, title, cost):

self.title = title

self.cost = cost

self.count = 1

def \_\_str\_\_(self):

return f"{self.title} (Count: {self.count}) - £{self.cost:.2f}"

class Basket:

def \_\_init\_\_(self):

self.basket = [] # List to store items in the basket

self.costs = [] # List to store costs for each item in the basket

# ... rest of your code ...

def display\_available\_items(self):

print("\nAvailable Items:")

# Display available items with corresponding keys

for key, item in AVAILABLE\_ITEMS.items():

print(f"{key}. {item.title} - £{item.cost:.2f}")

return AVAILABLE\_ITEMS

# ... rest of your code ...

**After Improvement 2: Renaming Variables and Functions**

class PetItem:

def \_\_init\_\_(self, title, cost):

self.title = title

self.cost = cost

self.count = 1

def \_\_str\_\_(self):

return f"{self.title} (Count: {self.count}) - £{self.cost:.2f}"

class Basket:

def \_\_init\_\_(self):

self.items\_in\_basket = [] # List to store items in the basket

self.costs = [] # List to store costs for each item in the basket

def add\_item\_to\_basket(self, item, count):

# Check if the item is already in the basket

basket\_item = next((basket\_item for basket\_item in self.items\_in\_basket if basket\_item.title == item.title), None)

if basket\_item:

# If the item is in the basket, update the count

basket\_item.count += count

print(f"Added {count} more '{item.title}' to your basket.")

else:

# If the item is not in the basket, add it

item.count = count

self.items\_in\_basket.append(item)

print(f"Added {count} {item.title} to your basket.")

self.costs.extend([item.cost] \* count)

# ... rest of your code ...

Remember to replace all instances of the old variable and function names with the new ones throughout your code. Also, make sure to test your code after making these changes to ensure everything still works as expected. If you encounter any issues or have any questions, feel free to ask! I’m here to help. 😊