

Project Report:

Portfolio Milestone Module 4 – Online Shopping Cart

Alexander Ricciardi

Colorado State University Global

CSC500: Principles of Programming

Professor: Dr. Brian Holbert

October 5, 2025

Project Report:

Portfolio Milestone Module 4 – Online Shopping Cart

This documentation is part of the Portfolio Milestone Module 4 from CSC500: Principles of Programming at Colorado State University Global. This Project Report is an overview of the program's functionality and testing scenarios, including console output screenshots. The program is coded in Python 3.13, and it is called Portfolio Milestone Module 4 – Online Shopping Cart.

The Assignment Direction:

Online Shopping Cart

Step 1: Build the ItemToPurchase class with the following specifications:

- Attributes
- item_name (string)
- item_price (float)
- item_quantity (int)
- Default constructor
- Initializes item's name = "none", item's price = 0, item's quantity = 0
- Method
- print_item_cost()

Example of print_item_cost() output:

Bottled Water 10 @ \$1 = \$10

Step 2: In the main section of your code, prompt the user for two items and create two objects of the ItemToPurchase class.

Example:

Item 1

Enter the item name:

Chocolate Chips

Enter the item price:

3

Enter the item quantity:

1

Item 2

Enter the item name:

Bottled Water

Enter the item price:

1

Enter the item quantity:

10

Step 3: Add the costs of the two items together and output the total cost.

Example:

TOTAL COST

Chocolate Chips 1 @ \$3 = \$3

Bottled Water 10 @ \$1 = \$10

Total: \$13

Your program submission materials must include your source code and screenshots of the application executing the code and the results. Please refer to the video as a recourse and reference: [Python Classes and Objects \(With Examples\)](#).

My Program Description:

The program is a small terminal app. it is an implementation of an online shopping cart implementation. The program renders banners and a menu, and it calculates the total cost of items based on each item's price and quantity. Only two items are asked to be entered in this implementation.

⚠ My notes:

As I was using some of the same code lines for my critical assignments, I created several small utilities functions and classes that I can reuse; they are found below the comments:

```
# _____
# General Utility Functions
#
```

And

```
# _____
# Utility Classes
#
```

Git Repository:

I use [GitHub](#) as my Distributed Version Control System (DVCS).

The following is a link to my GitHub profile, [Omega.py](#).

The link to this specific assignment is:

<https://github.com/Omegapy/My-Academics-Portfolio/tree/main/MS-in-AI-Machine-and-Learning/CSC500-Principles-of-Programming/Critical-Thinking-Module-3>

Figure-1

Code on GitHub

My-Academics-Portfolio/MS-in-AI-Machine-and-Learning / CSC500-Principles-of-Programming / Portfolio-Milestone-Module-4 / online_shopping.cart.py

Omegapy Adds the Portfolio-Milestone-Module-4 project to the My-Academics-Portfolio repository

Code Blame 549 lines (475 loc) - 19.7 KB

```
1 # -----
2 # File: online_shopping.cart.py
3 # Project:
4 # Author: Alexander Ricciardi
5 # Date: 2025-09-01
6 # File Path: Portfolio-Milestone-Module-4/online_shopping.cart.py
7 # -----
8 # Course: CSC-500 Principles of Programming
9 # Professor: Dr. Brian Holbert
10 # Fall C-2025
11 # Sep-Nov 2025
12 # -----
13 # Assignment:
14 # Portfolio Milestone Module 4
15 # -----
16 # Directions:
17 # Online Shopping Cart
18 #
19 # Step 1: Build the ItemToPurchase class with the following specifications:
20 # * Attributes
21 #   * item_name (string)
22 #   * item_price (float)
23 #   * item_quantity (int)
24 #   * Default constructor
25 #   * Initializes item's name = "none", item's price = 0, item's quantity = 0
26 # * Method
27 #   * print_item_cost()
28 # Example of print_item_cost() output:
29 # Bottled Water 10 @ $1 = $10
30 #
31 # Step 2: In the main section of your code, prompt the user for two items and create two objects of the ItemToPurchase class.
32 # Examples:
33 # Item 1
34 # Enter the item name:
35 # Chocolate Chips
36 # Enter the item price:
37 # 3
38 # Enter the item quantity:
39 # 1
40 #
41 # Item 2
42 # Enter the item name:
43 # Bottled Water
44 # Enter the item price:
45 # 1
46 # Enter the item quantity:
47 # 10
48 #
49 # Step 3: Add the costs of the two items together and output the total cost.
50 # Example:
51 # TOTAL COST
52 # Chocolate Chips 1 @ $3 = $3
53 # Bottled Water 10 @ $1 = $10
54 # Total: $13
55 #
56 # -----
57 # Project:
58 # Online Shopping Cart
59 #
60 # Project description:
61 # The program is a small terminal app. that implements the functionality of
62 # an online shopping cart.
63 #
64 #
65 #
66 # ---- Module Contents Overview ---
67 # - Class: Banner
68 # - Class: Menu
69 # - Class: ItemToPurchase
70 # - Function: validate_prompt_int()
71 # - Function: validate_prompt_float()
72 # - Function: validate_prompt_string()
73 # - Function: main()
74 #
75 #
76 # ---- Dependencies / Imports ---
77 # - Standard library: dataclasses, decimal, typing
78 # - Requirements ---
79 # - Python 3.13
80 #
81 #
82 # ---- Apache 2.0 ---
83 # © 2025 Alexander Samuel Ricciardi - All rights reserved.
84 # License: Apache-2.0 | Code
85 #
86 #
87 #
88 # Online shopping cart implementation
89 #
90 # The program is a small terminal app.
91 # it is an implementation of an online shopping cart implementation
92 # The program renders banners and a menu,
93 # and it calculates the total cost of items based on each item's price and quantity.
94 # Only two items are asked to be entered in this implementation.
95 #
96 #
97 #
98 # Imports
99 #
100 #
101 from dataclasses import dataclass
102 from decimal import Decimal
103 from typing import Any, Literal, Sequence, TypeAlias
104 #
105 #
106 # -----
107 # ||| Added/extra features not part of the assignment |||
108 # ||| See Steps 1, 2, and 3 for assignment requirements |||
109 # ||| -----
110 #
111 #
112 #
113 #
114 #
115 #
116 def validate_prompt_yes_or_no(prompt: str) -> bool:
117     """Prompt the user for a yes/no confirmation"""
118     while True:
119         choice = input(f"(y/n): {prompt}").lower()
120         if choice in ["y", "n"]:
121             return choice == "y"
122         else:
123             print("Please enter 'y' or 'n'.")
```

Code Snippet 1*Project Code in VS Code*

```
# -----
# File: online_shopping_cart.py
# Project:
# Author: Alexander Ricciardi
# Date: 2025-09-29
# [File Path] Portfolio-Milestone-Module-4/online_shopping_cart.py
# -----
# Course: CSS-500 Principles of Programming
# Professor: Dr. Brian Holbert
# Fall C-2025
# Sep-Nov 2025
# -----
# Assignment:
# Portfolio Milestone Module 4
#
# Directions:
# Online Shopping Cart
#
# Step 1: Build the ItemToPurchase class with the following specifications:
#   • Attributes
#   • item_name (string)
#   • item_price (float)
#   • item_quantity (int)
#   • Default constructor
#   • Initializes item's name = "none", item's price = 0, item's quantity = 0
#   • Method
#   • print_item_cost()
# Example of print_item_cost() output:
# Bottled Water 10 @ $1 = $10
#
# Step 2: In the main section of your code, prompt the user for two items and create two
# objects of the ItemToPurchase class.
# Example:
#   Item 1
#   Enter the item name:
#   Chocolate Chips
#   Enter the item price:
#   3
#   Enter the item quantity:
#   1
#
#   Item 2
#   Enter the item name:
```

```
# Bottled Water
# Enter the item price:
# 1
# Enter the item quantity:
# 10
#
# Step 3: Add the costs of the two items together and output the total cost.
# Example:
# TOTAL COST
# Chocolate Chips 1 @ $3 = $3
# Bottled Water 10 @ $1 = $10
# Total: $13

# -----
# Project:
# Online Shopping Cart
#
# Project description:
# The program is a small terminal app. that implements the functionality of
# an online shopping cart.
#
# -----
#
# --- Module Contents Overview ---
# - Class: Banner
# - Class: Menu
# - Class: ItemToPurchase
# - Function: validate_prompt_int()
# - Function: validate_prompt_float()
# - Function: validate_prompt_string()
# - Function: main()
#
# -----
#
# --- Dependencies / Imports ---
# - Standard Library: dataclasses, decimal, typing
# --- Requirements ---
# - Python 3.13
#
# -----
#
# --- Apache-2.0 ---
# © 2025 Alexander Samuel Ricciardi - All rights reserved.
# License: Apache-2.0 | Code
#
# -----
#
#####
#####
```

```

Online shopping cart implementation

The program is a small terminal app.
it is an implementation of an online shopping cart implementation
The program renders banners and a menu,
and it calculates the total cost of items based on each item's price and quantity.
Only two items are asked to be entered in this implementation.

"""

# -----
# Imports
#
from dataclasses import dataclass
from decimal import Decimal
from typing import Any, Literal, Sequence, TypeAlias

# =====
# ||| Added/extr features not part of the assignment |||
# ||| See Steps 1, 2, and 3 for assignment requirements |||
# ||| =====

# -----
# General Utility Functions
#
# ----- validate_prompt_yes_or_no()
def validate_prompt_yes_or_no(prompt: str) -> bool:
    """Prompt the user for a yes/no confirmation"""
    while True:
        choice = input(f"{prompt} [Y/N]: ").strip().lower()
        if choice in ("y", "yes"):
            return True
        if choice in ("n", "no"):
            return False
        print("Invalid input. Please enter 'Y' or 'N'.") # ----- end validate_prompt_yes_or_no()

# ----- wait_for_enter()
def wait_for_enter() -> None:
    """Terminal pause: Waits until user presses Enter"""
    input("\nPress Enter to continue...")
# ----- end wait_for_enter()

```

```

# _____
# User input validation utility functions
#
# ----- validate_prompt_int()

def validate_prompt_int(prompt: str) -> int:
    """Prompt until a valid integer is entered"""
    while True:
        raw = input(f"{prompt}\n").strip()
        try:
            return int(raw)
        except ValueError:
            print("Invalid input. Please enter a whole integer (e.g., 3).")

# ----- end validate_prompt_int()

# ----- validate_prompt_float()

def validate_prompt_float(prompt: str) -> float:
    """Prompt User until a valid float is entered"""
    while True:
        raw = input(f"{prompt}\n").strip()
        try:
            return float(raw)
        except ValueError:
            print("Invalid input. Please enter a valid float (e.g., 12.99).")

# ----- end validate_prompt_float()

# ----- validate_prompt_string()

def validate_prompt_string(prompt: str) -> str:
    """Prompt User until a valid float is entered"""
    while True:
        raw = input(f"{prompt}\n").strip()
        try:
            if raw == "": raise ValueError()
            return str(raw)
        except ValueError:
            print("Invalid input. Please enter a string (e.g., Hello).")

# ----- end validate_prompt_string()

#
# Utility Classes
#
#

```

```

# Terminal Banner
#
# -----
# ----- class Banner
class Banner:
    """Render Banner object to be on the terminal

    It renders a banner box using box-drawing characters using text alignments
    left/center/right

    """

    Alignment: TypeAlias = Literal["left", "center", "right"]

    #
    # Class constants
    #
    DEFAULT_TEXT: str = "Banner"
    DEFAULT_ALIGNMENT: str = "center"
    DEFAULT_WIDTH: int = 44

    #
    # -----
    # Constructor
    #
    # -----
    def __init__(
        self,
        text: str = DEFAULT_TEXT,
        alignment: Alignment = DEFAULT_ALIGNMENT,
        width: int = DEFAULT_WIDTH,
    ) -> None:
        """Initialize a new banner

        Args:
            text: Text in the banner
            alignment: Alignment for the text (left/center/right)
            width: width of the banner content area
        """
        self._lines: list[tuple[str, Alignment]] = [(text, alignment)]
        self.width: int = max(width, len(text), 3)

    # -----
    # ----- end __init__()

    #
    # -----
    # Private Methods
    #

```

```

# ----- _from_lines()
@classmethod
def _from_lines(
    cls,
    lines: Sequence[tuple[str, Alignment]],
    width: int = DEFAULT_WIDTH,
) -> "Banner":
    """Construct a banner using lines of box-drawing characters

Args:
    lines: Sequence of text/alignment
    width: inner width of the banner

Returns:
    "Banner" to be render

Raises:
    ValueError: if lines: is empty
"""

if not lines:
    raise ValueError("Banner.from_lines requires at least one line.")
first_text, first_align = lines[0]
inst = cls(text=first_text, alignment=first_align, width=width)
inst._lines = list(lines)
content_width = max(len(t) for t, _ in inst._lines)
inst.width = max(width, content_width, 3)
return inst
# ----- end _from_lines()

#
# -----
# Banner constructor helper methods
#

def _top(self) -> str:
    return f"{'=' * self.width}\n"

def _divider(self) -> str:
    return f"{'=' * self.width}\n"

def _bottom(self) -> str:
    return f"{'=' * self.width}\n"

def _text_line(self, text: str, alignment: Alignment = "center") -> str:
    if alignment == "left":

```

```

        return f"{{text.ljust(self.width)}}"
    if alignment == "right":
        return f"{{text.rjust(self.width)}}"
    return f"{{text.center(self.width)}}"

# -----
# Render banner
#
# ----- render()
def render(self) -> str:
    """Render the banner text as a single string that can be displayed

    Returns:
        The rendered banner in the form of a string
    """
    output = [self._top()]
    for text, align in self._lines:
        output.append(self._text_line(text, align))
    output.append(self._bottom())
    return "\n".join(output)
# ----- end render()

# ----- end class Banner

# -----
# Terminal app Menu
#
# ----- class Menu
class Menu:
    """Render a Menu object to be on the terminal using the Banner class"""

    # ----- __init__()
    def __init__(
        self,
        title: str,
        options: Sequence[str],
        width: int = Banner.DEFAULT_WIDTH,
    ) -> None:
        """Create a new menu

        Args:
            title: Menu title
            options: Menu options, sequence starting at 1.
            width: inner Banner width

```

```

    Raises:
        ValueError: if options is empty.
    """
    if not options:
        raise ValueError("Menu requires at least one option.")
    self.title = title
    self._options = list(options)
    self._width = width
    self._numbered_options = [
        f"{index}. {label}" for index, label in enumerate(self._options, start=1)
    ]
    self._banner_lines = [
        (self.title, "center"),
        *[(option, "left") for option in self._numbered_options],
    ]
    self._banner = Banner._from_lines(self._banner_lines, width=self._width)
# ----- end __init__()

# _____
# Menu constructor helper methods
#
# def _choices(self) -> list[str]:
#     """Turn index choices number to strings (e.g., ["1", "2"])."""
#     return [str(index) for index in range(1, len(self._options) + 1)]

def _choice_index_range(self) -> str:
    """Turn choice (str) index into a displayable range (e.g., "1-3" or "1")
    see _choices()
    """
    options = self._choices()
    if len(options) == 1:
        return options[0] # "1"
    return f"{options[0]}-{options[-1]}" # (e.g., "1-3")

def _choice_index_list(self) -> str:
    """Lists available index choices for errors message (e.g., "1, 2, or 3")."""
    options = self._choices()
    if len(options) == 1:
        return options[0]
    return ", ".join(options[:-1]) + f", or {options[-1]}"

# ----- render()
def render(self) -> str:
    """Render the menu as a banner string

```

```

    Returns:
        A string of the fully form Menu
    """
sections = [
    self._banner._top(),
    self._banner._text_line(self.title, alignment="center"),
    self._banner._divider(),
]
sections.extend(
    self._banner._text_line(option, alignment="left") for option in
                                                self._numbered_options
)
sections.append(self._banner._bottom())
return "\n".join(sections)

# ----- end render()

# ----- end class Menu

# =====
# ||| Step 1: Build the ItemToPurchase class |||
# ||
# -----
# App Classes Definitions
#
# ----- class ItemToPurchase
@dataclass
class ItemToPurchase:
    """Represent a purchasable item with its name, price, and quantity.

    This is the minimal domain model used by the console backend to compute
    line-item and total costs. It does not perform I/O and holds only data
    relevant to pricing and quantities.
    """
#
# Assignment requirement
# -- Step 1 Initializes item variables
item_name: str = "none"
item_price: float = 0.0
item_quantity: int = 0

#
# Assignment requirement

```

```

# -- Step 1 part of the print_item_cost() output format
# ----- format_currency()
@staticmethod
def format_currency(value: float) -> str:
    """Return a formatted float (e.g., 10.00 = 10) remove unnecessary trailing zero
    Example:
        2.0 -> "2"; 2.5 -> "2.5"; 2.75 -> "2.75".
    """
    if value == int(value):
        return str(int(value))
    return f"{value:.2f}".rstrip("0") # Strip unnecessary 0s
# ----- end format_currency()

# -----
# Assignment requirement
# -- Step 1 print_item_cost() method
# ----- print_item_cost()
def print_item_cost(self) -> str:
    """Print and return cost for this item

    Returns:
        a cost line (e.g., "Apples 3 @ $1 = $3")
    """
    total_cost = self.item_price * self.item_quantity
    cost_statement = (
        f"{self.item_name} {self.item_quantity} @ "
        f"${self.format_currency(self.item_price)} = "
        f"${self.format_currency(total_cost)}"
    )
    print(cost_statement)
    return cost_statement
# ----- end print_item_cost()

# ----- end class ItemToPurchase

# -----
# ----- Main Function -----
#
# ----- main()
def main() -> None:
    """Run the shopping cart program"""
    #
    # -----
    # Embedded Helper Functions
    #

```

```

# ----- build_banner()
def build_banner(text: str) -> str:
    """Create a banner string

Args:
    text: The text inside the banner box

Returns:
    A string, the full banner, with borders
"""

banner = Banner(text=text, alignment="center", width=Banner.DEFAULT_WIDTH)
return banner.render()
# ----- end build_banner()

# ----- prompt_for_item()
def prompt_for_item(item_number: int) -> ItemToPurchase:
    """Prompt the user a new item values
    and from it creates a ItemToPurchase` object

Args:
    item_number: the item sequenced number of creation

Returns:
    ItemToPurchase object instance with name, price, and quantity entered

Raises:
    ValueError: if user input is not valide
"""

print(f"Item {item_number}")
item_name = validate_prompt_string("Enter the item name:")
item_price = validate_prompt_float("Enter the item price:")
item_quantity = validate_prompt_int("Enter the item quantity:")
return ItemToPurchase(item_name=item_name, item_price=item_price,
                     item_quantity=item_quantity)

# ----- end prompt_for_item()

# -----
# Instance Banner and Menu objects
#
HEADER = build_banner("Online Shopping Cart")
ENTRY_BANNER = build_banner("Provide the item information")
RESULTS_BANNER = build_banner("TOTAL COST")
MAIN_MENU = Menu("Menu", ["Calculate total for two items", "Exit"])

```

```
print(HEADER)

# _____
# Main Loop
#
while True:
    print()
    print(MAIN_MENU.render())
    choice = input(f"Select an option ({MAIN_MENU._choice_index_range()}): ").strip()

    if choice == "1":
        # =====
        # ||| Step 2: prompt the user for two items and create
        # ||| two objects of the ItemToPurchase class
        # |||
        # =====
        print()
        print(ENTRY_BANNER)
        first_item = prompt_for_item(1)
        print()
        second_item = prompt_for_item(2)
        # =====
        # ||| Step 3: Add the costs of the two items together
        # ||| and output the total cost
        # |||
        # =====
        total_cost = (
            first_item.item_price * first_item.item_quantity
            + second_item.item_price * second_item.item_quantity
        )

        print()
        print(RESULTS_BANNER)
        first_item.print_item_cost()
        second_item.print_item_cost()
        print(f"Total: ${ItemToPurchase.format_currency(total_cost)}")
        wait_for_enter()

    elif choice == "2":
        if (validate_prompt_yes_or_no("Are you sure that you want to exist? ")):
            print("\nBye! 🙏\n")
            break

    else:
```

```
    print(f"\nInvalid selection. Please enter {MAIN_MENU._choice_index_list()}.")\n\n# ----- end main()\n#\n# _____\n# Module Initialization / Main Execution Guard (if applicable)\n#\n# ----- main_guard\nif __name__ == "__main__":\n    main()\n# ----- end main_guard\n#\n# _____\n# End of File\n#
```

Continue next page

Figure 2

Project Outputs in the VS Code Terminal

```

PS P:\CSC-500\Portfolio-Milestone-Module-4> uv run online_shopping_cart.py
[Online Shopping Cart]
[Menu]
1. Calculate total for two items
2. Exit

Select an option (1-2): 1

Provide the item information

Item 1
Enter the item name:
Chocolate Chips
Enter the item price:
3
Enter the item quantity:
1

Item 2
Enter the item name:
Bottled Water
Enter the item price:
1
Enter the item quantity:
10

[TOTAL COST]

Chocolate Chips 1 @ $3 = $3
Bottled Water 10 @ $1 = $10
Total: $13

Press Enter to continue...

[Menu]
1. Calculate total for two items
2. Exit

Select an option (1-2): 2
Are you sure that you want to exist? [Y/N]: y
Bye! 🖐

PS P:\CSC-500\Portfolio-Milestone-Module-4>

```

The screenshot shows the execution of a Python script named `online_shopping_cart.py` in a VS Code terminal. The terminal interface includes a sidebar with various icons (problems, output, terminal, etc.) and a status bar at the bottom.

The script starts by displaying a menu with options 1 and 2. The user selects option 1, which prompts for item information. Two items are entered: Chocolate Chips at \$3 each and Bottled Water at \$1 each, with quantities of 1 and 10 respectively. The total cost is calculated and displayed as \$13.

After the calculation, the user is given the option to exit. They respond with 'y', indicating they want to exit. The script then says "Bye!" and exits.

Figure 3*Project Outputs Troubleshooting in the VS Code Terminal*

```

PS P:\CSC-500\Portfolio-Milestone-Module-4> uv run online_shopping_cart.py
[Terminal]
Online Shopping Cart

[Terminal]
Menu
1. Calculate total for two items
2. Exit

Select an option (1-2): 3
Invalid selection. Please enter 1, or 2.

[Terminal]
Menu
1. Calculate total for two items
2. Exit

Select an option (1-2): w
Invalid selection. Please enter 1, or 2.

[Terminal]
Menu
1. Calculate total for two items
2. Exit

Select an option (1-2): 1
Provide the item information

Item 1
Enter the item name:
Invalid input. Please enter a string (e.g., Hello).
Enter the item name:
Banana
Enter the item price:
2.00
Invalid input. Please enter a valid float (e.g., 12.99).
Enter the item price:
2.45
Enter the item quantity:
1.4
Invalid input. Please enter a whole integer (e.g., 3).
Enter the item quantity:
12

Item 2
Enter the item name:
Lemon
Enter the item price:
$2333.55
Invalid input. Please enter a valid float (e.g., 12.99).
Enter the item price:
012.00
Enter the item quantity:
10

TOTAL COST

Banana 12 @ $2.45 = $29.4
Lemon 10 @ $12 = $120
Total: $149.4

Press Enter to continue...

```

```

    Menu
1. Calculate total for two items
2. Exit

Select an option (1-2): 1

Provide the item information

Item 1
Enter the item name:
Lemon
Enter the item price:
1.55
Enter the item quantity:
1

Item 2
Enter the item name:
Banana
Enter the item price:
1.23456
Enter the item quantity:
10

TOTAL COST

Lemon 1 @ $1.55 = $1.55
Banana 10 @ $1.23 = $12.35
Total: $13.9

Press Enter to continue...

    Menu
1. Calculate total for two items
2. Exit

Select an option (1-2): 2
Are you sure that you want to exist? [Y/N]: f
Invalid input. Please enter 'Y' or 'N'.
Are you sure that you want to exist? [Y/N]: n

    Menu
1. Calculate total for two items
2. Exit

Select an option (1-2): 2
Are you sure that you want to exist? [Y/N]: y
Bye! 🌟

PS P:\CSC-500\Portfolio-Milestone-Module-4> []

<main* 0 11 0 0 0> LF {} Python Finish Setup 3.13.7 (csc-500) 6:03 PM
Search 9/29/2025

```

As shown in Figures 2, 3, and Code Snippet 1, the program runs without any issues, displaying the correct outputs as expected.