Video Game Wishlist Manager

Mark Dedvukaj

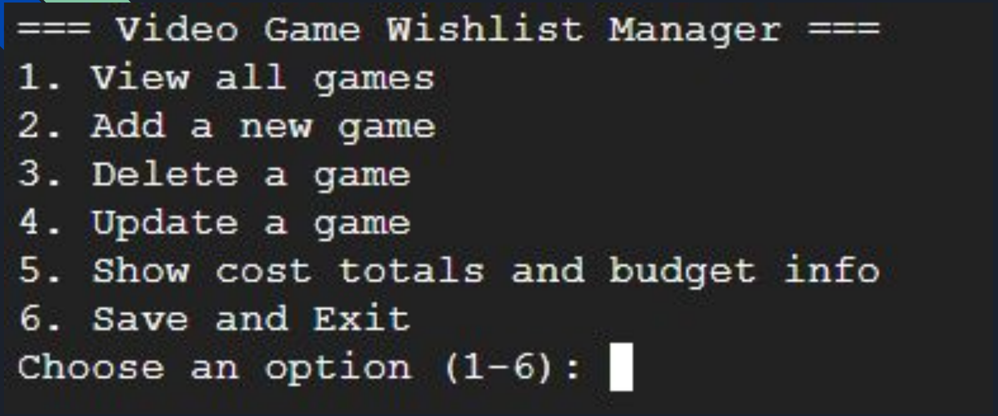
Problem

Some people want to track what videogames they want to buy or that they already have. This program solves that problem of keeping track of video games and their costs using a simple python interface.

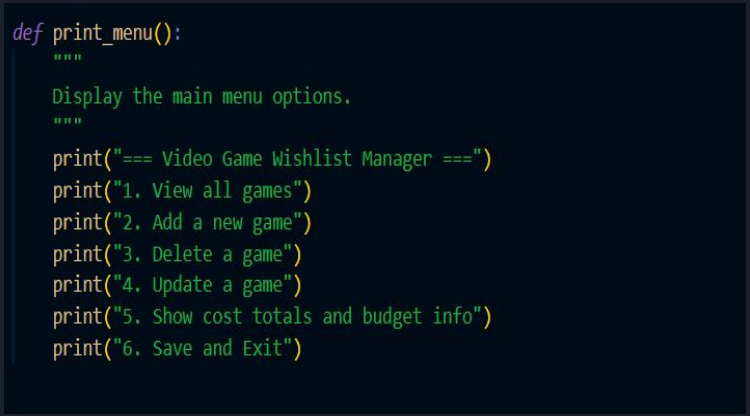
Structure

- `load_games()` This reads the file
- `save_games()` This writes the file
- Other functions:
  - `add_game()`
  - `delete_game()`
  - `update_game()`
  - `show_budget_info()`
  - `calculate_total_cost()`
- Main loop handles menu options

Main menu



```
=== Video Game Wishlist Manager ===
1. View all games
2. Add a new game
3. Delete a game
4. Update a game
5. Show cost totals and budget info
6. Save and Exit
Choose an option (1-6):
```

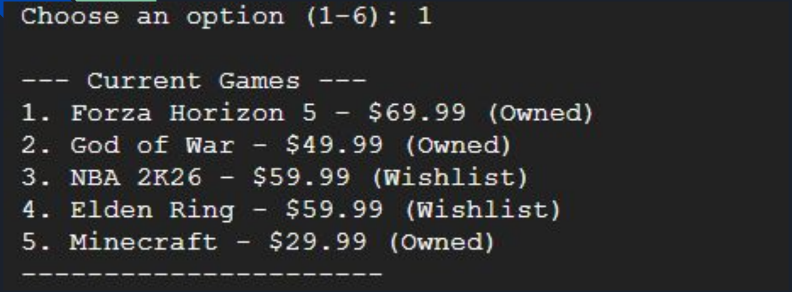


```
def print_menu():
    """
    Display the main menu options.
    """
    print("=== Video Game Wishlist Manager ===")
    print("1. View all games")
    print("2. Add a new game")
    print("3. Delete a game")
    print("4. Update a game")
    print("5. Show cost totals and budget info")
    print("6. Save and Exit")
```

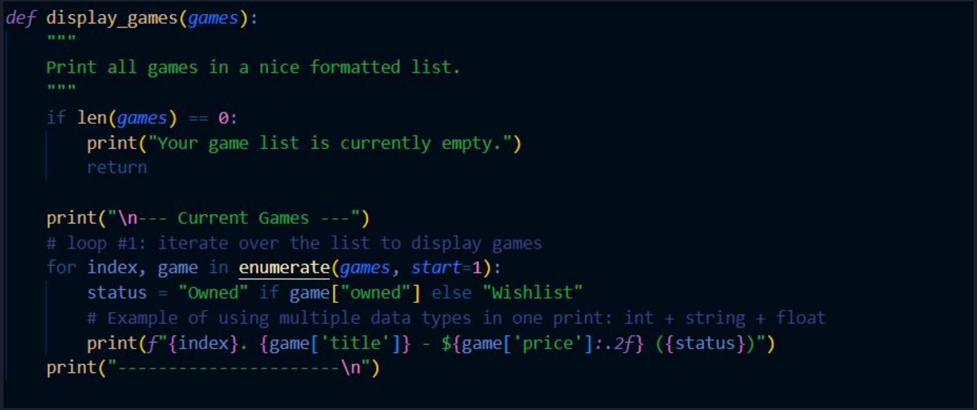
My video game wishlist manager has 6 parts and they can be accessed by user input. Option 1 allows you to view games, 2 allows you to add more games, 3 deletes games, 4 updates the games, 5 shows the total cost, and 6 saves and exits the program.

I used print commands for this section to display each menu option.

View all games



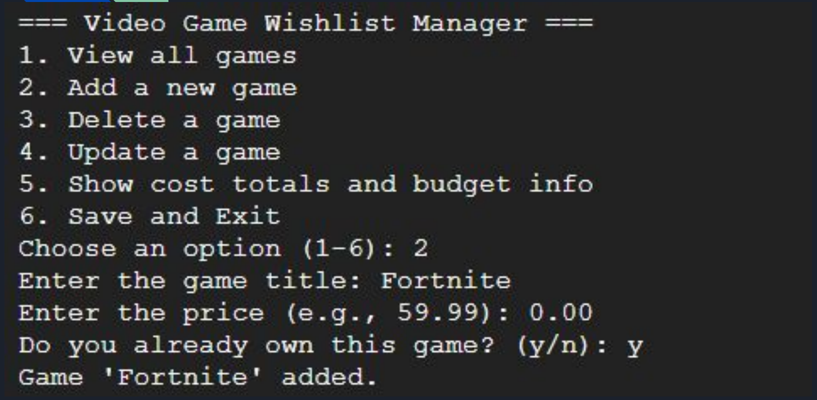
```
Choose an option (1-6): 1

--- Current Games ---
1. Forza Horizon 5 - $69.99 (Owned)
2. God of War - $49.99 (Owned)
3. NBA 2K26 - $59.99 (Wishlist)
4. Elden Ring - $59.99 (Wishlist)
5. Minecraft - $29.99 (Owned)
----------------------
```



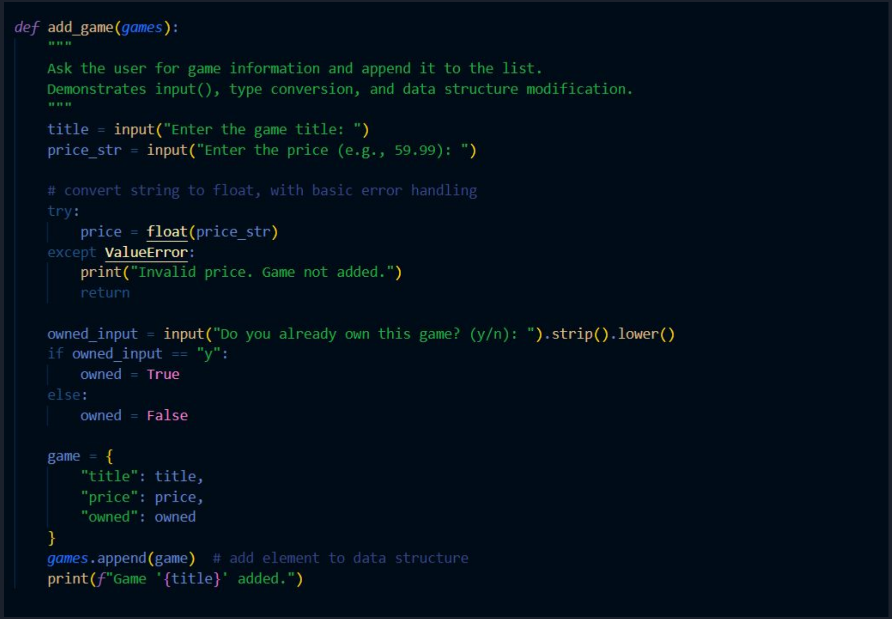
```python
def display_games(games):
    """
    Print all games in a nice formatted list.
    """
    if len(games) == 0:
        print("Your game list is currently empty.")
        return

    print("\n--- Current Games ---")
    # loop #1: iterate over the list to display games
    for index, game in enumerate(games, start=1):
        status = "Owned" if game["owned"] else "Wishlist"
        # Example of using multiple data types in one print: int + string + float
        print(f"{index}. {game['title']} - ${game['price']:.2f} ({status})")
    print("----------------------\n")
```

If you want to view all games in the wishlist type in 1 and it will show you all the current games you own or want to own. For this part to work properly you need to upload you games.txt file into python so it will show the games.

Add a New Game



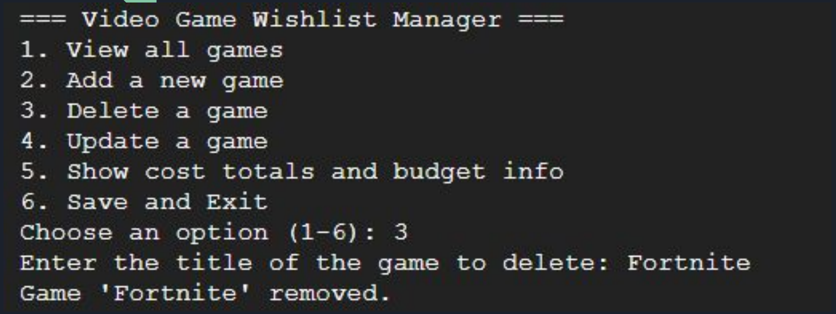
```
=== Video Game Wishlist Manager ===
1. View all games
2. Add a new game
3. Delete a game
4. Update a game
5. Show cost totals and budget info
6. Save and Exit
Choose an option (1-6): 2
Enter the game title: Fortnite
Enter the price (e.g., 59.99): 0.00
Do you already own this game? (y/n): y
Game 'Fortnite' added.
```



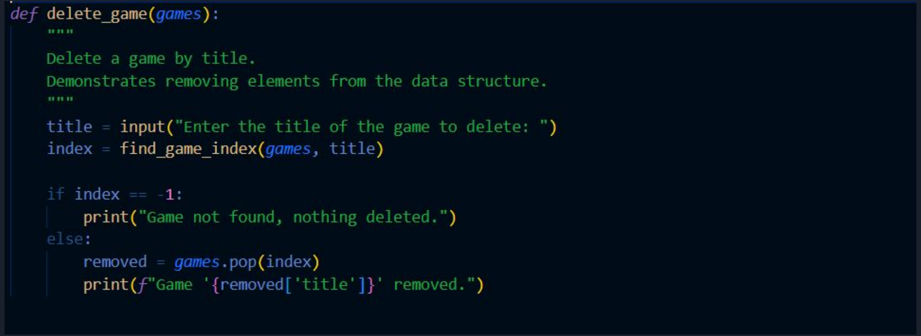
```python
def add_game(games):
    """
    Ask the user for game information and append it to the list.
    Demonstrates input(), type conversion, and data structure modification.
    """
    title = input("Enter the game title: ")
    price_str = input("Enter the price (e.g., 59.99): ")

    # convert string to float, with basic error handling
    try:
        price = float(price_str)
    except ValueError:
        print("Invalid price. Game not added.")
        return

    owned_input = input("Do you already own this game? (y/n): ").strip().lower()
    if owned_input == "y":
        owned = True
    else:
        owned = False

    game = {
        "title": title,
        "price": price,
        "owned": owned
    }
    games.append(game)  # add element to data structure
    print(f"Game '{title}' added.")
```

Option 2 allows you to add a new game. When you press 2 it will ask you for game name, price, and whether you own the game or not. After adding this information it will add the game to your wishlist.

Delete a Game



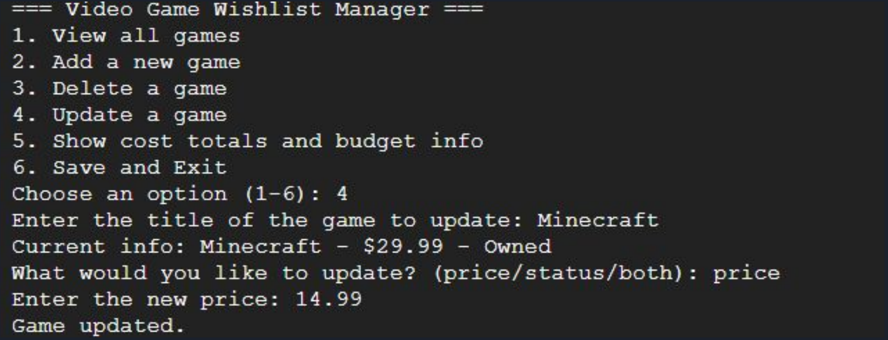
```
=== Video Game Wishlist Manager ===
1. View all games
2. Add a new game
3. Delete a game
4. Update a game
5. Show cost totals and budget info
6. Save and Exit
Choose an option (1-6): 3
Enter the title of the game to delete: Fortnite
Game 'Fortnite' removed.
```



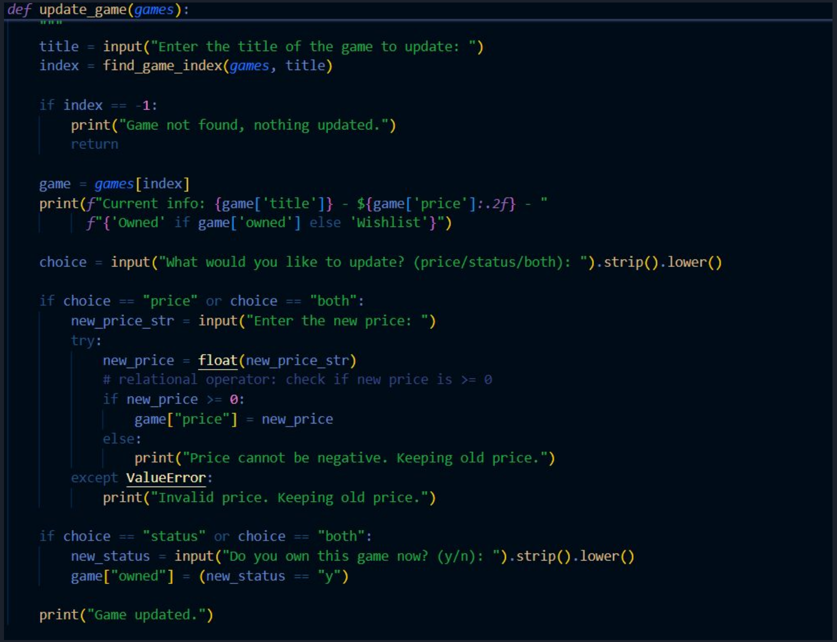
```python
def delete_game(games):
    """
    Delete a game by title.
    Demonstrates removing elements from the data structure.
    """
    title = input("Enter the title of the game to delete: ")
    index = find_game_index(games, title)

    if index == -1:
        print("Game not found, nothing deleted.")
    else:
        removed = games.pop(index)
        print(f"Game '{removed['title']}' removed.")
```

Option 3 allows you to delete a game. When you want to delete a game it will ask you the name of the game you want to delete and it will delete it.

Update a Game



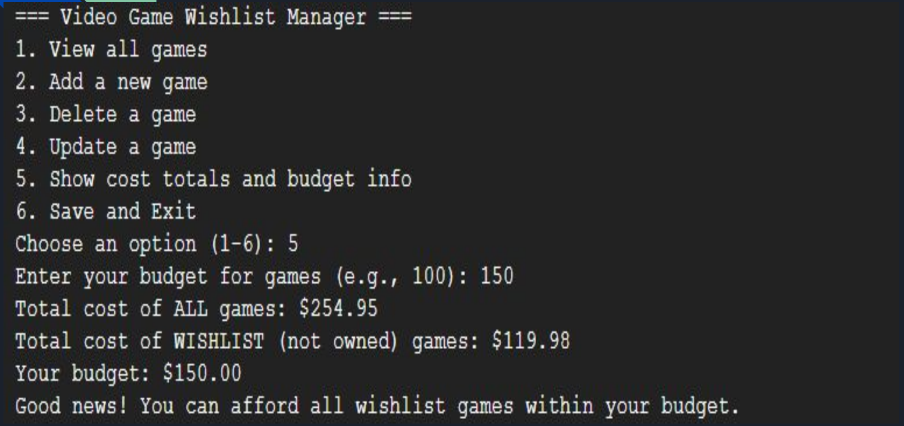
```
=== Video Game Wishlist Manager ===
1. View all games
2. Add a new game
3. Delete a game
4. Update a game
5. Show cost totals and budget info
6. Save and Exit
Choose an option (1-6): 4
Enter the title of the game to update: Minecraft
Current info: Minecraft - $29.99 - Owned
What would you like to update? (price/status/both): price
Enter the new price: 14.99
Game updated.
```



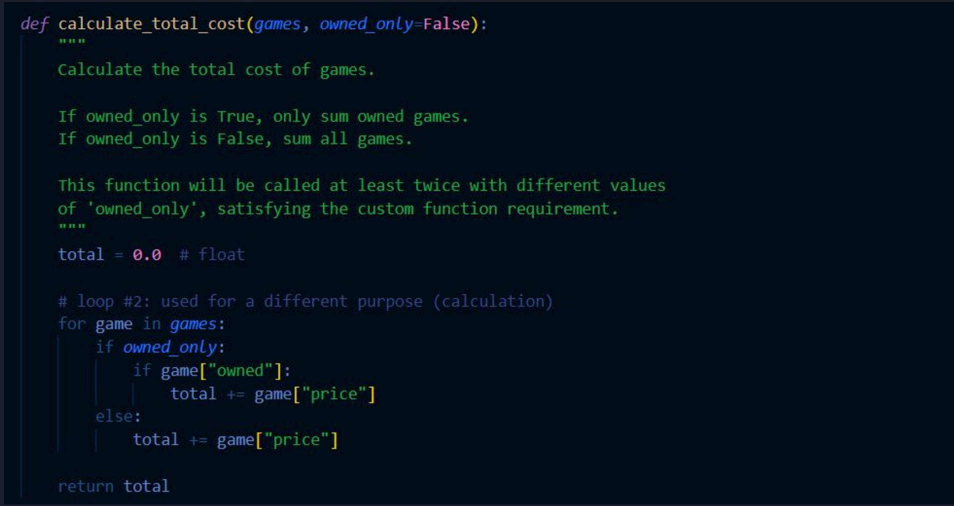
```python
def update_game(games):
    """
    title = input("Enter the title of the game to update: ")
    index = find_game_index(games, title)

    if index == -1:
        print("Game not found, nothing updated.")
        return

    game = games[index]
    print(f"Current info: {game['title']} - ${game['price']:.2f} - "
          f"{'Owned' if game['owned'] else 'Wishlist'}")

    choice = input("What would you like to update? (price/status/both): ").strip().lower()

    if choice == "price" or choice == "both":
        new_price_str = input("Enter the new price: ")
        try:
            new_price = float(new_price_str)
            # relational operator: check if new price is >= 0
            if new_price >= 0:
                game["price"] = new_price
            else:
                print("Price cannot be negative. Keeping old price.")
        except ValueError:
            print("Invalid price. Keeping old price.")

    if choice == "status" or choice == "both":
        new_status = input("Do you own this game now? (y/n): ").strip().lower()
        game["owned"] = (new_status == "y")

    print("Game updated.")
```

Option 4 allows you to update a game. When updating a game you can update the price, if you own or want the game, or you can do both. After updating the game when you go back to option 1 you will see all of the updates you did.

Show Cost Totals and Budget Information



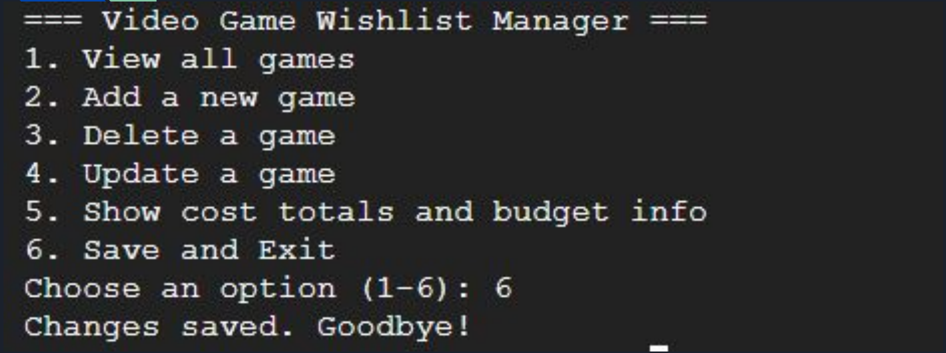
```
=== Video Game Wishlist Manager ===
1. View all games
2. Add a new game
3. Delete a game
4. Update a game
5. Show cost totals and budget info
6. Save and Exit
Choose an option (1-6): 5
Enter your budget for games (e.g., 100): 150
Total cost of ALL games: $254.95
Total cost of WISHLIST (not owned) games: $119.98
Your budget: $150.00
Good news! You can afford all wishlist games within your budget.
```



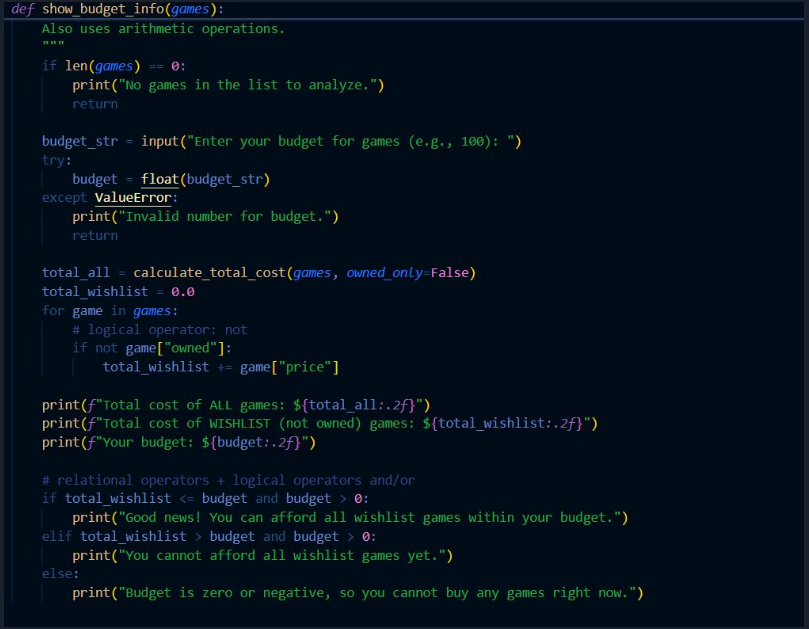
```python
def calculate_total_cost(games, owned_only=False):
    """
    Calculate the total cost of games.

    If owned_only is True, only sum owned games.
    If owned_only is False, sum all games.

    This function will be called at least twice with different values
    of 'owned_only', satisfying the custom function requirement.
    """
    total = 0.0  # float

    # loop #2: used for a different purpose (calculation)
    for game in games:
        if owned_only:
            if game["owned"]:
                total += game["price"]
        else:
            total += game["price"]

    return total
```

Option 5 allows you to input a value that corresponds to your budget. Lets say your budget is $150, that covers the cost of all the games in the wishlist. The code adds the prices off all the games together to show you the value of all the games.

Save and Exit



```
=== Video Game Wishlist Manager ===
1. View all games
2. Add a new game
3. Delete a game
4. Update a game
5. Show cost totals and budget info
6. Save and Exit
Choose an option (1-6): 6
Changes saved. Goodbye!
```



```python
def show_budget_info(games):
    Also uses arithmetic operations.
    """
    if len(games) == 0:
        print("No games in the list to analyze.")
        return

    budget_str = input("Enter your budget for games (e.g., 100): ")
    try:
        budget = float(budget_str)
    except ValueError:
        print("Invalid number for budget.")
        return

    total_all = calculate_total_cost(games, owned_only=False)
    total_wishlist = 0.0
    for game in games:
        # logical operator: not
        if not game["owned"]:
            total_wishlist += game["price"]

    print(f"Total cost of ALL games: ${total_all:.2f}")
    print(f"Total cost of WISHLIST (not owned) games: ${total_wishlist:.2f}")
    print(f"Your budget: ${budget:.2f}")

    # relational operators + logical operators and/or
    if total_wishlist <= budget and budget > 0:
        print("Good news! You can afford all wishlist games within your budget.")
    elif total_wishlist > budget and budget > 0:
        print("You cannot afford all wishlist games yet.")
    else:
        print("Budget is zero or negative, so you cannot buy any games right now.")
```

Option 6 saves your changes and exits the program.