Project 4-1: Even or Odd Checker

Create a program that checks whether a number is even or odd.

Console

```
Even or Odd Checker

Enter an integer:  33
This is an odd number.
```

Specifications

- Store the code that gets user input and displays output in the main() function.
- Store the code that checks whether the number is even or odd in a separate function.
- Assume that the user will enter a valid integer.

Project 4-2: Hike Calculator

Create a program that converts the number of miles that you walked on a hike to the number of feet that you walked.

Console

```
Hike Calculator

How many miles did you walk?: 4.5
You walked 23760 feet.
```

Specifications

- The program should accept a float value for the number of miles.
- Store the code that displays the title in a separate function.
- Store the code that converts miles to feet in a separate function. This function should return an int value for the number of feet.
- There are 5280 feet in a mile.
- Store the code that gets user input and displays output in the main() function.
- Assume that the user will enter a valid number of miles.

Project 4-3: Feet and Meters Converter

Create a program that uses a separate module to convert feet to meters and vice versa.

Console

```
Feet and Meters Converter

Conversions Menu:
a. Feet to Meters
b. Meters to Feet
Select a conversion (a/b): a

Enter feet: 100
30.48 meters

Would you like to perform another conversion? (y/n): y

Conversions Menu:
a. Feet to Meters
b. Meters to Feet
Select a conversion (a/b): b

Enter meters: 100
328.08 feet

Would you like to perform another conversion? (y/n): n

Thanks, bye!
```

Specifications

- The formula for converting feet to meters is:

  ```
  feet = meters / 0.3048
  ```

- The formula for converting meters to feet is:

  ```
  meters = feet * 0.3048
  ```

- Store the code that performs the feet to meters and meters to feet conversions in functions within a module.
- Store the code that displays the title in its own function, and store the code that displays the menu in its own function, but store the rest of the code that gets input and displays output in the main() function.
- Assume the user will enter valid data.
- The program should round results to a maximum of two decimal places.

Project 4-4: Sales Tax Calculator

Create a program that uses a separate module to calculate sales tax and total after tax.

Console

```
Sales Tax Calculator

ENTER ITEMS (ENTER 0 TO END)
Cost of item: 35.99
Cost of item: 27.50
Cost of item: 19.59
Cost of item: 0
Total:           83.08
Sales tax:       4.98
Total after tax: 88.06

Again? (y/n): y

ENTER ITEMS (ENTER 0 TO END)
Cost of item: 152.50
Cost of item: 59.80
Cost of item: 0
Total:           212.3
Sales tax:       12.74
Total after tax: 225.04

Again? (y/n): n

Thanks, bye!
```

Specifications

- The sales tax rate should be 6% of the total.
- Store the sales tax rate in a module. This module should also contain functions that calculate the sales tax and the total after tax. These functions should round the results to a maximum of two decimal places.
- Store the code that gets input and displays output in the main() function. Divide this code into functions whenever you think it would make that code easier to read and maintain.
- Assume the user will enter valid data.

Project 4-5: Dice Roller

Create a program that uses a function to simulate the roll of a die.

Console

```
Dice Roller

Roll the dice? (y/n): y

Die 1: 3
Die 2: 6
Total: 9

Roll again? (y/n): y

Die 1: 1
Die 2: 1
Total: 2
Snake eyes!

Roll again? (y/n): y

Die 1: 6
Die 2: 6
Total: 12
Boxcars!

Roll again? (y/n): n
```

Specifications

- The program should roll two six-sided dice.
- Store the code that rolls a single die in a function.
- Store the code that gets input and displays output in the main() function. Divide this code into functions whenever you think it would make that code easier to read and maintain.
- The program should display a special message for two ones (snake eyes) and two sixes (boxcars).

Project 4-6: Prime Number Checker

Create a program that checks whether a number is a prime number and displays the total number of factors if it is not a prime number.

Console

```
Prime Number Checker

Please enter an integer between 1 and 5000: 1
Invalid integer. Please try again.
Please enter an integer between 1 and 5000: 2
2 is a prime number.

Try again? (y/n): y

Please enter an integer between 1 and 5000: 3
3 is a prime number.

Try again? (y/n): y

Please enter an integer between 1 and 5000: 4
4 is NOT a prime number.
It has 3 factors.

Try again? (y/n): y

Please enter an integer between 1 and 5000: 6
6 is NOT a prime number.
It has 4 factors.

Try again? (y/n): n

Bye!
```

Specifications

- A prime number is only divisible by two factors (1 and itself). For example, 7 is a prime number because it is only divisible by 1 and 7.
- If the number is not a prime number, the program should display its number of factors. For example, 6 has four factors (1, 2, 3, and 6).
- Store the code that gets a valid integer for this program in its own function.
- Store the code that calculates the number of factors for a number in its own function.
- Store the rest of the code that gets input and displays output in the main() function. Divide this code into functions whenever you think it would make that code easier to read and maintain.