

1. You're building a simple calculator tool that helps users perform safe division. It should take a number from the user and divide 10 by that number, handling common input errors gracefully.

Write a **program** that asks the user to enter a number and then prints the result of dividing 10 by that number. The program should handle two types of errors:

- If there is a **ValueError**, print "Please enter a valid number."
- If there is a **ZeroDivisionError**, print "Cannot divide by zero."

**Examples:**

- "Enter a number:" 5 → 2.0
  - "Enter a number:" 0 → "Cannot divide by zero."
  - "Enter a number:" "hello" → "Please enter a valid number."
2. You're building a simple menu selector for a fruit delivery service. Customers can choose a fruit by entering its index from a list of available options.

Write a **program** that asks the user to enter an index and then prints the corresponding item from a predefined list, shown below:

- ['apple', 'banana', 'cherry', 'date']

The program should handle two types of errors:

- If the input is not a valid number (**ValueError**), print "Invalid index format."
- If the number is out of range (**IndexError**), print "Index out of range."

**Examples:**

- "Enter an index:" 1 → "banana"
  - "Enter an index:" 5 → "Index out of range."
  - "Enter an index:" "two" → "Invalid index format."
3. You're building a shopping assistant that helps users look up the prices of products from a catalog.

Write a **program** that asks the user to enter the name of a product and then prints its price from a predefined dictionary, shown below:

- {'apple': 1.5, 'banana': 0.9, 'cherry': 2.2}

The program should handle two types of issues:


- If the product is not found in the dictionary **KeyError**, print "Product not found."
- If the input is empty, print "Please enter a product name."  
Hint: this will not produce an error. Handle it with logic.

**Examples:**

- "Enter product name:" "apple" → 1.5

- “Enter product name:” “mango” → “Product not found.”
  - “Enter product name:” “” → “Please enter a product name.”
4. Write a **program** that asks the user to enter the name of a text file and then prints the contents of that file to the screen. The program should handle the following error:
- If the file does not exist (**FileExistsError**), print “File not found.”

**Examples:**

<input type="checkbox"/> Name	Date modified	Type
 contents	4/17/2025 ...	Text Document
 LunchData	4/17/2025 ...	Text Document

- “Enter file name:” “LunchData.txt” → (prints the content of LunchData.txt)
  - “Enter file name:” “DinnerData.txt” → “File not found.”
5. You are helping a teacher update students’ scores after a quiz. The teacher wants to add points for extra credit and needs your program to do the math safely.

A dictionary stores the number of points each student has earned, shown below:

- {‘Alice’: 90, ‘Bob’: 75, ‘Charlie’: 60}

Write a **program** that asks the user to enter a student’s name and a number to add to their score. The program should print the new number of points.

The program should handle the following errors:

- If the name is not found in the dictionary (**KeyError**), print “Student not found.”
- If the number entered is not valid (e.g., not a number) (**ValueError**), print “Invalid number.”

**Examples:**

- “Enter student name:” “Bob”  
“Enter number to subtract:” 10 → 65
  - “Enter student name:” “David” → “Student not found.”
  - “Enter student name:” “Alice”  
“Enter number to subtract:” “ten” → “Invalid number.”
6. You’re building a simple scheduling tool that lets users select a day of the week by entering a number between 0 and 6. Each number corresponds to a day, starting with 0 for Monday and ending with 6 for Sunday.

A list contains the days of the week, shown below:

- [‘Monday’, ‘Tuesday’, ‘Wednesday’, ‘Thursday’, ‘Friday’, ‘Saturday’, ‘Sunday’].

Write a **program** that asks the user to enter a number (0–6) and prints the corresponding day.

The program should handle the following errors:

- If the input is not a valid number (**ValueError**), print “Invalid input.”

- If the number is outside the valid range (**IndexError**), print “Index out of range.”

**Examples:**

- “Enter a number:” 0  $\rightarrow$  “Monday”
- “Enter a number:” 6  $\rightarrow$  “Sunday”
- “Enter a number:” 7  $\rightarrow$  “Index out of range.”
- “Enter a number:” “two”  $\rightarrow$  “Invalid input.”

7. You're building a tool that compares two numbers by calculating both the difference and the ratio. The program should ask the user to enter two numbers and then:

- Print the difference (first minus second)
- Print the result of dividing the first number by the second

Write a **program** that uses `int()` to convert user input and performs both calculations.

The program should handle the following errors:

- If either input is not a valid number (**ValueError**), print "Invalid input."
- If the second number is 0 (**ZeroDivisionError**), print "Cannot divide by zero."
- If the result is too large (**OverflowError**), print "Result too large."

**Examples:**

- [illegible]

8. You are building a color picker feature for a drawing app. The user selects a color by entering its position in a preset list of colors.

A list contains some colors, shown below:

- ['red', 'green', 'blue', 'yellow', 'purple']

Write a **program** that asks the user to enter an **index** and then prints the corresponding color. If the user enters an invalid index or input, they should be asked to try again until a valid index is given.

The program should handle the following errors:

- If the input is not a number (**ValueError**), print "Invalid input. Try again."
- If the index is out of range (**IndexError**), print "Index out of range. Try again."

**Examples:**

- “Enter an index:” 2  $\rightarrow$  “blue”
- “Enter an index:” 10  $\rightarrow$  “Index out of range. Try again.”
- “Enter an index:” “green”  $\rightarrow$  “Invalid input. Try again.”
- (after retry) “Enter an index:” 1  $\rightarrow$  “green”

9. You've been asked to help build part of a travel booking system. One of the features lets users type in a country code (like "US") and shows them the full country name to confirm their destination.

A dictionary stores some country codes, shown below:

```
{'US':'United States', 'FR':'France', 'JP':'Japan', 'BR':'Brazil'}
```

Write a **program** that asks the user to enter a country code (like "US") and then prints the full country name. If the user enters an invalid code, they should be asked to try again until a valid code is entered.

The program should handle the following errors:

- If the code is not found in the dictionary (**KeyError**), print "Code not found. Try again."

**Examples:**

- "Enter a country code:" "JP" → "Japan"
- "Enter a country code:" "XYZ" → "Code not found. Try again."
- (after retry) "Enter a country code:" "BR" → "Brazil"

10. A game show awards a cash prize to be split evenly among a group of winners. Write a **program** that asks the user to:

- (a) Enter the prize amount in dollars
- (b) Enter the number of winners

The program should calculate how much each person gets by dividing the prize by the number of winners. The program should keep asking until the input is valid, and handle the following errors:

- If the input is not a number (e.g. typing "five") (**ValueError**), print "Invalid input. Try again."
- If there are no winners (**ZeroDivisionError**), print "Must have at least one winner. Try again."

**Examples:**

- "Enter prize amount:" 10000  
"Enter number of winners:" 4 → 2500.0
- "Enter number of winners:" 0 → "Cannot divide by zero. Try again."
- "Enter prize amount:" "grand" → "Invalid input. Try again."