

# Tutorial V - Handling Errors

## Programming with Python

### Basic exception handling

We start with a simple task to get you familiar with the concept of exception handling. You are given an empty function that takes three numbers as input. It adds the first two numbers and then divides the result by the third number. You have to use a try-except block to handle the ZeroDivisionError.

```
# TODO: Write a function that takes three numbers as input. It adds the
first two numbers and then divides the result by the third number. Use a
try-except block to handle the ZeroDivisionError.
def safe_divide(add_1, add_2, div):
    # Your code here
    pass

# Test cases
print(safe_divide(5, 5, 2)) # Should print: 5.0
print(safe_divide(10, 0, 0)) # Should print: "Error: Division by zero"
```

### Handling Multiple Exceptions

In this exercise, you'll have to handle multiple exceptions in one try-except block, as we also want to catch exceptions with a wrong type of input, e.g. when the user inputs a string instead of a number.

```
# a) TODO: Modify the previous function to handle both ZeroDivisionError
and TypeError
def safe_divide_v2(add_1, add_2, div):
    # Your code here
    pass

# Test cases
print(safe_divide_v2(5, 5, 2)) # Should print: 5.0
print(safe_divide_v2(10, 0, 0)) # Should print: "Error: Division by zero"
print(safe_divide_v2(2, 4, "2")) # Should print: "Error: Invalid input
types"

# b) TODO: Write a function that asks the user for a number and then
divides it by a second number inputted by the user.
# - Use a try-except block to handle the exceptions.
# - Use a while loop to repeatedly ask the user for a number and divide it
by a second number until the user inputs "no" to the question "Do you want
to continue?".
```

## Raising your own exceptions

In this exercise, you'll have to raise your own exceptions when the user inputs a wrong type of input, e.g. when the user inputs a string instead of a number. Your task is to write a function that asks the user a username and then checks if the username is valid. A valid username is considered to be a number that is at least 5 characters long and contains no spaces. If the username is not valid, you should raise an exception, tell the user that the username is not valid and ask for a new username. You should only accept the username if it is valid.

```
# TODO: Write a function that asks the user for a username and then checks
if the username is valid.
# - A valid username is considered to be a number that is at least 5
characters long and contains no spaces.
# - If the username is not valid, you should raise an exception, tell the
user that the username is not valid and ask for a new username.
# - You should only accept the username if it is valid.

# You can start by changing the code from the lecture:
class InvalidUsernameError(Exception):
    pass

def get_valid_username():
    while True:
        try:
            username = input("Please enter a username (no spaces): ")
            if " " in username:
                raise InvalidUsernameError("Username must not contain
spaces.")
            return username
        except InvalidUsernameError as e:
            print(f"Invalid username: {e}")
            print("Please try again.")
```

## Using Assertions

By using assertions, we can check if the input of a function is correct. If the assertion is not correct, an `AssertionError` is raised. This is especially useful in the development phase to catch errors that should not occur.

```
# TODO: Write a function that calculates the area of a rectangle. Ensure
that the length and width are positive numbers.
def calculate_rectangle_area(length, width):
    # Your code here
    pass

# Test cases
print(calculate_rectangle_area(5, 3))    # Should print: 15
print(calculate_rectangle_area(-5, 3))   # Should raise AssertionError
print(calculate_rectangle_area(5, "3"))  # Should raise AssertionError
```

## Debugging

In the following exercise, you'll have to debug a function that is supposed to return the sum of all even numbers in a list. However, there is a bug in the code. Can you find it and fix it? Use either print statements, assertions, or an IDE's debugger to fix the code.

```
# TODO: Fix the bug in the following function.
def sum_even_numbers(numbers):
    total = 0
    for num in numbers:
        if num % 2 == 0:
            total + num
    return total

# Test case
print(sum_even_numbers([1, 2, 3, 4, 5, 6])) # Should print: 12, but it's
not working correctly

# Bonus challenge: Add error handling to make this function more robust
```

## Anagram Scramble Game

In this final exercise, you'll create an anagram game that combines everything you've learned about error handling, functions, and file I/O. The computer will scramble a word from a file, and the player must unscramble it within a limited number of tries.

### Game Rules:

1. Load words from the file `words.txt`
2. Computer randomly selects and scrambles a word
3. Player has 5 tries to guess the original word
4. Player can type "quit" to end the game
5. Display success or the correct answer at the end

### Your Tasks:

```
import random

# TODO: Implement the function to load words from file
def load_words(filename):
    """
    Load words from a file (one word per line).

    Returns a list of words (converted to lowercase and stripped).
    Handle FileNotFoundError - print error message and return empty list.
    """
    # Your code here
    pass

# TODO: Implement word scrambling function
def scramble_word(word):
    """
```

```

    Scramble the letters of a word randomly.

    Hint: Convert to list, use random.shuffle(), convert back to string.
    Make sure the scrambled word is different from the original.
    """
    # Your code here
    pass

# TODO: Implement input validation
def validate_guess(guess, original_word):
    """
    Validate that a guess is acceptable.

    - Raise ValueError if guess is empty or contains only spaces
    - Raise ValueError if guess contains numbers or special characters
    (use .isalpha())
    - Raise ValueError if guess length doesn't match original word
    - Return True if guess matches original word (case-insensitive), False
    otherwise
    """
    # Your code here
    pass

# TODO: Implement the main game function
def play_anagram_game():
    """
    Main game function that orchestrates the entire game.

    1. Load words from 'words.txt'
    2. If no words loaded, print error and return
    3. Randomly select a word using random.choice()
    4. Scramble the word
    5. Give player 5 attempts
    6. In a loop:
        - Show scrambled word and remaining tries
        - Get player input
        - Handle "quit" to exit
        - Validate the guess using validate_guess()
        - Catch and handle all exceptions with appropriate messages
        - Check if guess is correct
        - Break loop if correct or out of tries
    7. Display win message or reveal the correct word
    """
    # Your code here
    pass

# Test the game
if __name__ == "__main__":
    play_anagram_game()

```

## Requirements Checklist:

- ☐ Load words from file with FileNotFoundError handling
- ☐ Scramble word randomly (ensure it's different from original)

- [?] Validate empty input (ValueError)
- [?] Validate input contains only letters (ValueError)
- [?] Validate guess length matches word (ValueError)
- [?] Player has exactly 5 tries
- [?] Handle “quit” command
- [?] Display clear feedback for each error type
- [?] Show win/lose message with correct word

### Tips:

- Use `random.choice(words)` to select a random word from the list
- Use `random.shuffle()` on a list of characters to scramble
- Use `.strip()` and `.lower()` for input processing
- Use `.isalpha()` to check if string contains only letters
- Use a while loop with a counter for tries
- Make sure scrambled word  $\neq$  original (use a while loop to keep shuffling if needed)
- Use try-except blocks in the main game loop to keep game running after errors

### That’s it!

After a week, you can find the solutions to these exercises online in the associated GitHub repository, but we will also quickly go over them in next week’s tutorial. To access the solutions, click on the Github button on the lower right and search for the folder with today’s lecture and tutorial. Alternatively, you can ask ChatGPT or Claude to explain them to you. Remember, the goal is not just to complete the exercises, but to understand the concepts and improve your programming abilities.