Lab Assignment
& Solution

Cybersecurity Professional Program

Introduction to Python
for Security

File System & Error Handling

PY-04-LS2
Error Handling

Note: Solutions for the instructor are shown inside the green box.

Lab Objective

Understand how error detection and handling controls code execution.

Lab Mission

Practice handling error conditions that may occur in Python code.

Formatted: Left

Lab Duration

15–20 minutes

Requirements

- Working knowledge of basic programming.
- Working knowledge of exception handling.

Resources

- Environment & Tools
  - Windows
    - PyCharm
    - Python3

Textbook References

- Chapter 4: File System and Error Handling
  - Section 1: Error Handling

Lab Task: Product Calculation in Python

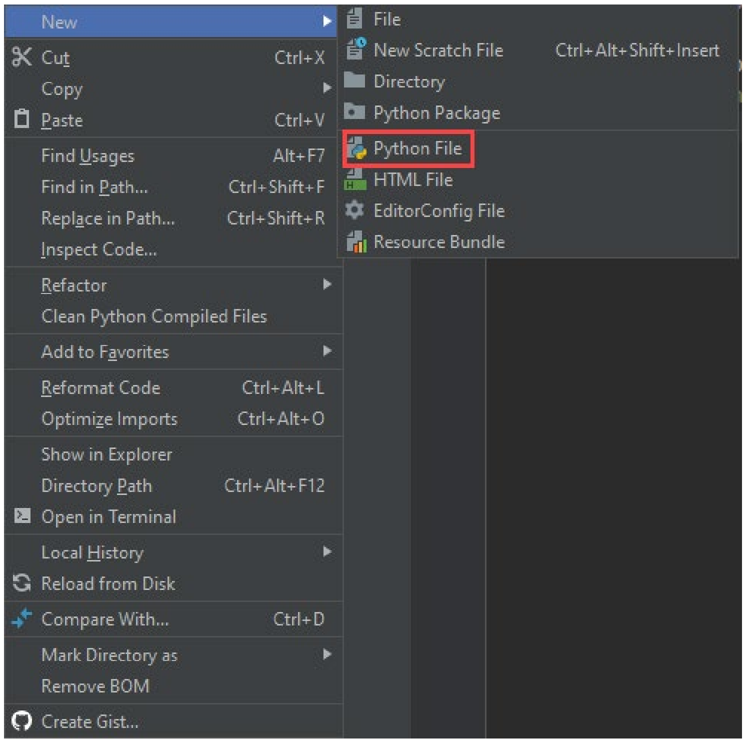
Write a program that calculates the product of four numbers provided by the user~~,~~ and prints the result. Use '***try***' and '***except***' statements to ensure that the program will not fail when the input is not a valid number.

Formatted: Left

1 Create a new Python file in PyCharm by right-clicking the project you created and selecting **New** ~~>~~→ ***Python File***.

Formatted: Indent: Left: 0.13", Hanging: 0.38"

Formatted: Font: Italic



2 Declare the variable '***product***' and assign it an integer value of *1*.

Formatted: Indent: Left: 0.13", Hanging: 0.38"

Formatted: Font: Italic

```
product = 1
```

3 Create a ***'for'*** loop that performs four iterations.

Formatted: Left, Indent: Left: 0.13", Hanging: 0.38"

```
product = 1
for i in range(4):
```

4 In the ***for*** loop, ask the user to provide a number, cast that number to an integer, and assign it to a new variable. Multiply each input by the ***'product'*** variable~~,~~ and assign the result to the same variable.

Formatted: Left, Indent: Left: 0.13", Hanging: 0.38"

Formatted: Font: Bold, Italic

```
product = 1
for i in range(4):
    num = int(input("Enter a number: "))
    product *= num
```

5 Place the user input and mathematical operation in a ***"try"*** block. Make sure that you use indentation when you place it in the ***"try"*** block.

Formatted: Left, Indent: Left: 0.13", Hanging: 0.38"

```
product = 1
for i in range(4):
    try:
        num = int(input("Enter a number: "))
        product *= num
```

6 Create an ***'except'*** block that prints a message to the console if the user inputs anything other than a number.

Formatted: Left, Indent: Left: 0.13", Hanging: 0.38"

```
product = 1
for i in range(4):
    try:
        num = int(input("Enter a number: "))
        product *= num
    except :
        print("The input is not a valid number")
```

7 Run the code from S~~s~~tep ~~5,~~6 input an integer, then input a non-integer, and observe the results.

Formatted: Indent: Left: 0.13", Hanging: 0.38"

```
Enter a number: 4
Enter a number: 3
Enter a number: 2
Enter a number: A
The input is not a valid character
```

When a non-integer is entered as the last input, the invalidation message will appear, as it does in the example below, because the non-integer input is caught by the ~~'~~***except***~~'~~ statement in the ~~'~~***try***~~'~~ block.

Formatted: Left, Indent: Left: 0.5"

8 At the bottom of the script, print a message that tells the user the product of the four numbers and cast the ~~"~~***total***~~"~~ variable to a string.

Formatted: Left, Indent: Left: 0.13", Hanging: 0.38"

9 Rerun the code, but this time input integers only. Observe the results.

```
product = 1
for i in range(4):
    try:
        num = int(input("Enter a number: "))
        product~~total~~ *= num
    except:
        print("The input is not a valid number")
print("The product of the 4 numbers is: "~~ ~~+ str(product~~total~~))
===================================================================
Enter a number: 4
Enter a number: 3
Enter a number: 2
Enter a number: 1
The product of the 4 numbers is: 24
```