4CS001

Python Workshop 11: Exception Handling

Part 1

1. How many except statements can a try-except block have?
   a) zero
   b) one
   c) one or more
   d) none of the above
2. When will the else part of try-except-else be executed?
   a) always
   b) when an exception occurs
   c) when no exception occurs
   d) when an exception occurs in to except block
3. Is the following Python code valid?

```
try:
    # Do something
except:
    # Do something
finally:
    # Do something
```

   a) no, there is no such thing as finally
   b) no, finally cannot be used with except
   c) no, finally must come before except
   d) yes
4. Is the following Python code valid?

```
try:
    # Do something
except:
    # Do something
else:
    # Do something
```

   a) no, there is no such thing as else
   b) no, else cannot be used with except
   c) no, else must come before except
   d) yes
5. When is the finally block executed?
   a) when there is no exception
   b) when there is an exception
   c) only if some condition that has been specified is satisfied
   d) always

Part 2

1. Using try…except, showcase the ZeroDivisionError.

```
>>> try:
...     a = int(input("Enter a numerator: "))
...     b = int(input("Enter a denominator: "))
...     c = a / b
... except ZeroDivisionError:
...     print("Error: Division by zero")
... else:
...     print("The result is", c)
...
Enter a numerator: 9
Enter a denominator: 0
Error: Division by zero
>>>
```

2. Create a simple list containing five elements and try to print the sixth element of the list. [use IndexError exception]

```
>>> my_list = [1, 2, 3, 4, 5]
>>>
>>> try:
...     print("The sixth element of the list is:", my_list[5])
... except IndexError:
...     print("Error: Index out of range")
...
Error: Index out of range
>>>
```

3. Try printing a variable without declaring it first. [use NameError exception]

```
>>> try:
...     print(my_variable)
... except NameError:
...     print("Error: Variable not declared")
...
Error: Variable not declared
>>>
```

4. The 'else' in try…except…else statements is used to run the code on the else block if there are no exceptions in the 'except' block. Show an example.

```
>>> try:
...     a = int(input("Enter a numerator: "))
...     b = int(input("Enter a denominator: "))
...     c = a / b
... except ZeroDivisionError:
...     print("Error: Division by zero")
... else:
...     print("The result is", c)
...
Enter a numerator: 8
Enter a denominator: 2
The result is 4.0
```

5. In Python, we can choose to throw an exception if a condition occurs. To throw the exception, we use 'raise' keyword. Show an example.

```
>>> def divide(a, b):
...     if b == 0:
...         raise ZeroDivisionError("division by zero")
...     return a / b
...
>>> try:
...     result = divide(10, 0)
... except ZeroDivisionError as error:
...     print("Error:", error)
... else:
...     print("The result is", result)
...
Error: division by zero
>>>
```

Part 3

1. Ask the user for the numerator and denominator value; and perform division. If the user enters a number, the program will evaluate and produce the result.

```
>>> try:
...     numerator = int(input("Enter the numerator: "))
...     denominator = int(input("Enter the denominator: "))
...     result = numerator / denominator
... except ValueError:
...     print("Error: Invalid input, please enter a number.")
... except ZeroDivisionError:
...     print("Error: Division by zero.")
... else:
...     print("The result is", result)
...
Enter the numerator: 8
Enter the denominator: 4
The result is 2.0
>>>
```

If the user enters a non-numeric value then, the try block will throw a ValueError exception, and we can catch that using a first catch block 'except ValueError' by printing the message 'Entered value is wrong'.

```
>>> try:
...     # some code that might raise an exception
...     value = int(input("Enter a number: "))
... except ValueError:
...     print("Entered value is wrong")
...
Enter a number: u
Entered value is wrong
>>>
```

And suppose the user enters the denominator as zero. In that case, the try block will throw a ZeroDivisionError, and we can catch that using a second catch block by printing

the message ‘Can’t divide by zero’.

```
>>> try:
...     numerator = int(input("Enter the numerator: "))
...     denominator = int(input("Enter the denominator: "))
...     division = numerator / denominator
... except ZeroDivisionError:
...     print("Can't divide by zero")
... except:
...     print("Invalid input")
... else:
...     print(f"The division result is: {division}")
...
Enter the numerator: 6
Enter the denominator: 5
The division result is: 1.2
>>>
```

2. Ask the user to enter an amount of money. In the try block, run a condition to check if the input value is less than 10 thousand; in which case raise a ValueError and print your message inside it. In the except block, catch the ValueError we previously raised and print the message inside it.

```
>>> try:
...     money = float(input("Enter an amount of money: "))
...     if money < 10000:
...         raise ValueError("Amount should be greater than or equal to 10,000.")
... except ValueError as ve:
...     print(ve)
...
Enter an amount of money: 8000
Amount should be greater than or equal to 10,000.
```

3. An EOFError is raised when built-in functions like input() hits an end-of-file condition (EOF) without reading any data. The file methods like readline() return an empty string when they hit EOF. Show an example.

```
>>> try:
...     user_input = input("Enter some text (or press Ctrl + D to trigger an EOF): ")
... except EOFError:
...     print("EOFError raised: End of file reached.")
... else:
...     print(f"You entered: {user_input}")
...
Enter some text (or press Ctrl + D to trigger an EOF): ^Dpramdip
You entered: ♦pramdip
>>> _
```