Q 1. What is the role of try and exception block?

Answer –

Try and except blocks in python are used for dealing with the exceptions and errors in our program.

When we think that lines of code can end in an error, we can write those lines of code under try block. The try block will look for any error occurrence in the lines of code. Whenever an exception occurs, the program flow is transferred to the except block.

Inside the except block we can deal with the exception and write the code for appropriate actions to be performed.

For a try block we can have multiple except blocks. Where, each except block handles different exception types.

Q 2. What is the syntax for a basic try-except block?

Answer –

```
try:
    # here we write the lines of code that we want to monitor for any error.
except:
    #write the lines of code to handle the code flow after the occurrence of the error in try block.
```

Q 3. What happens if an exception occurs inside a try block and there is no matching

except block?

Answer –

If there is no matching except block then the program execution will stop with an exception. However, if there is a bare except block, then the new exception will get handled by that except block.

Q 4. What is the difference between using a bare except block and specifying a specific exception type?

Answer –

Except blocks with specific exception types are used to deal with the specified exception type. We can write our code to deal with that particular exception in that except block. On the other hand the bare exception block is a generic one, which is used to handle any type of exception and may contain generic actions to be performed.

Q 5. Can you have nested try-except blocks in Python? If yes, then give an example.

Answer –

Yes, we can have nested try except blocks in python. Nested try catch blocks help us to stay inside the outer try block and continue with the code execution inside the out try block.

```
try:
    num1 = int(input("Enter the first number: "))
    num2 = int(input("Enter the second number: "))

    try:
        # Inner try block
        result = num1 / num2
        print("Result:", result)
    except ZeroDivisionError:
        print("Error: Division by zero occurred.")
    except ValueError:
        print("Error: Invalid input.")
    except Exception as e:
        print("An error occurred:", str(e))
```

```
except ValueError:
    print("Invalid input. Please enter valid integers.")
except Exception as e:
    print("An error occurred:", str(e))
```

Q 6. Can we use multiple exception blocks, if yes then give an example.

Answer –

We can use multiple exception blocks. Each exception block handles different exceptions. Each exception block have different lines of code to get a particular task done.

```
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))

try:
    # Inner try block
    result = num1 / num2
    print("Result:", result)
except ZeroDivisionError:
    print("Error: Division by zero occurred.")
except ValueError:
    print("Error: Invalid input.")
except Exception as e:
    print("An error occurred:", str(e))
```

Q 7. Write the reason due to which following errors are raised:

a. EOFError
b. FloatingPointError
c. IndexError
d. MemoryError
e. OverflowError
f. TabError
g. ValueError

Answer –

EOFError - This error is raised when we are reading the input stream and we reach the end of the file.

FloatingPointError - This exception has different types such as OverflowError, ZeroDivisionError, and ValueError.

OverflowError - The overflow error comes when the calculation exceeds maximum representable value.

IndexError - This error is raised when we try to access an invalid index of list/string. Which means we are accessing an index, which is non-existent for the particular list/string etc.

MemoryError - This error occurs when the program tries to use more memory than the available memory in the system.

TabError - This error is raised when we mix the use of tabs and spaces in python and the consistency is missing.

ValueError - This error is raised when the function argument is an inappropriate value.

Q 8. Write code for the following given scenario and add try-exception block to it.

a. Program to divide two numbers
b. Program to convert a string to an integer
c. Program to access an element in a list
d. Program to handle a specific exception
e. Program to handle any exception

Answer –

Program to divide two numbers

```
try:
    num1 = int(input("Enter 1st number"))
    num2 = int(input("Enter 2nd number"))

    div = num1/num2

    print("numbers ",num1,"and",num2,"division = ",div)

except ZeroDivisionError as e:
    print(e)
except Exception as e:
    print(e)
```

Program to convert a string to an integer

```
try:
    test_str = "123"
    test_int = int(test_str)
    print(test_int)
except Exception as e:
    print(e)
```

Program to access an element in a list

```
try:
    test_list =[1,2,3,4]
```

```
    print(test_list[5])
except IndexError:
    print("Index error occured")
```

Program to handle a specific exception

```
try:
    test_list =[1,2,3,4]
    print(test_list[5])
except IndexError:
    print("Index error occured")
```

Program to handle any exception

```
try:
    test_list =[1,2,3,4]
    print(test_list[5])
except Exception as e:
    print(e)
```