Assignment

PW SKILLS

Q1. What is an Exception in python? Write the difference between Exceptions and Syntax errors.

Answer.

An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. A syntax error is something caught by the compiler/interpreter and its incorrect use of the language itself.

Q2. What happens when an exception is not handled? Explain with an example.

Answer:

If the exception is not handled by an except clause, the exception is re-raised after the finally clause has been executed.

Example:

```
def divide(x, y):
    try:
        result = x / y
    except ZeroDivisionError:
        print("division by zero!")
    else:
        print("result is", result)
    finally:
        print("executing finally clause")

divide(2, 1)
result is 2.0
executing finally clause
divide(2, 0)
division by zero!
executing finally clause
divide("2", "1")
executing finally clause
```

Q3. Which Python statements are used to catch and handle exceptions? Explain with an example.

Answer:

The try and except block in Python is used to catch and handle exceptions. Python executes code following the try statement as a “normal” part of the program. The code that follows the except statement is the program’s response to any exceptions in the preceding try clause.

Example:

```
try:
    open("text.txt","r")
except Exception as e:
    print("There is some issue with the code",e)
```

Q4. Explain with an example:

a. try and else
b. finally
c. raise

Answer:

The **try** block lets you test a block of code for errors. And, The **else** block lets you execute code when there is no error.

Example:

```
try:
  print("Hello")
else:
  print("Nothing went wrong")
```

The **finally** block lets you execute code, regardless of the result of the try- and except blocks.

Example:

```
try:
  print(x)
except:
  print("Something went wrong")
finally:
  print("The 'try except' is finished")
```

The **raise** keyword is used to raise an exception.You can define what kind of error to raise, and the text to print to the user.

Example:

```
x = -1
if x < 0:
  raise Exception("Sorry, no numbers below zero")
```

Q5. What are Custom Exceptions in python? Why do we need Custom Exceptions? Explain with an example.

Answer: Custom exceptions are exception types you define yourself in your project. They basically inherit from the Exception base class and implement the three common constructors found in exception types.

Example:

```
class NegativeError(Exception):
   def __init__(self, data):
       self.data = data
try:
   x = int(input("Enter a number between positive integer: "))
   if x < 0:
       raise NegativeError(x)
except NegativeError as e:
   print("You provided {}. Please provide positive integer values only".format(e))
```

Q6. Create a custom exception class. Use this class to handle an exception.

Answer:

```
class validateage(Exception):
    def __init__(self,msg):
        self.msg=msg

def validate_age(age):
    if age < 0:
        raise validateage("The Age cannot be less than Zero")
    elif age > 200:
        raise validateage("The Entered age is too high")
    else:
        print("Entered age is a Valid age")

try:
    age= int(input("Enter the age:"))
    validate_age(age)
except validateage as e:
    print(e)
```

Output:

```
Enter the age: 800
The Entered age is too high
```