Untitled8

December 19, 2023

[ ]:
```
#QUESTION ANSWER  (1)

#What is Exception in python

#ANSWER= Errors detected during execution are called exceptions and are not␣
 ↪unconditionally fatal

#QUESTION= Write the differen,are between Exceptions and syntax errors

#ANSWER=

#EXCEPTION
"""An exception is an abnormal event that occurs during the execution of a␣
 ↪program.Exceptions are usually caused by runtime errors.such as trying to␣
 ↪access a file that does not exist. Exception are not syntax errors,
but they can still prevent the program from running if they are not handled␣
 ↪properly."""

#SYNTAX

"""A syntax error means there's an error in syntax such as misspelled keywords,
a missing punctuation character, a missing bracket,
a missing closing parenthesis or any errors in basic framing sequence of␣
 ↪characters or
tokens which is necessarily to be written in a particular programming language.
 ↪"""
```

[24]:
```
#QUESTION(2)
#ANSWER=
"when an exception is not handled properly they can still prevent the program␣
 ↪from running."
#EXAMPLE

try:
    f=open("text11.txt",'r')
    f.write("write into my file")
except Exception as e:
    print("this is my file",e)
```

```
this is my file [Errno 2] No such file or directory: 'text11.txt'
```

[26]:
```
#QUESTION(3)
#ANSWER
"try and except block are used to catch and handle exceptions."
#EXAMPLE
try:
    f=open("test2.txt",'r')
except Exception as e:
    print("this is my except block",e)
```

```
this is my except block [Errno 2] No such file or directory: 'test2.txt'
```

[28]:
```
#QUESTION(4)
#1) TRY AND ELSE
try:
    f=open("test6.txt",'r')
    f.write("this is my course")
except Exception as e:
    print("this is my except block",e)
else:
    print("this will be executed once your try will executed without errors")
```

```
this is my except block [Errno 2] No such file or directory: 'test6.txt'
```

[29]:
```
#2)FINALLY

try:
    f=open("text4.txt",'r')
    f.write("write something")
finally:
    print("finally will executed itself in any situation")
```

```
finally will executed itself in any situation
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[29], line 4
      1 #2)FINALLY
      3 try:
----> 4     f=open("text4.txt",'r')
      5     f.write("write something")
      6 finally:
```

```
File /opt/conda/lib/python3.10/site-packages/IPython/core/interactiveshell.py:
↪282, in _modified_open(file, *args, **kwargs)
    275 if file in {0, 1, 2}:
    276     raise ValueError(
    277         f"IPython won't let you open fd={file} by default "
    278         "as it is likely to crash IPython. If you know what you are␣
↪doing, "
    279         "you can use builtins' open."
    280     )
--> 282 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'text4.txt'
```

[30]:
```
#3)RAISE
class vaildage(Exception):

    def __init__(self,mes):
        self.mes=mes
```

[31]:
```
def validatedage(age):
    if age<0:
        raise vaildage("entered age is negative")
    elif age>200:
        raise vaildage("entered age is very very high")
    else:
        print("age is valid")
```

[32]:
```
try:
    age=int(input("enter your age"))
    validatedage(age)
except vaildage as e:
    print(e)
```

```
enter your age 45
```

```
age is valid
```

[36]:
```
#QUESTION(5)
#ANSWER
"""Regular classes that inherit from custom Exception make it very easy to␣
↪create our own custom Exception which can make our programs easier to follow␣
↪and more readable.
Custom Exception can make your code much more readable and robust, and reduce␣
↪the amount of code you write later to try and figure out what exactly went␣
↪wrong."""
```

```
#EXAMPLE

class validage(Exception):

    def __init__(self,mes):
        self.mes=mes
```

[37]:
```
def vailidae(age):
    if age<18:
        raise validage("not allow to enter our company")
    elif age >30:
        raise validage("Yes company to allow")
    else:
        print("allow to all in my company")
```

[38]:
```
try:
    age=int(input("enter our age"))
    vailidae(age)
except validage as e:
    print(e)
```

```
enter our age 67
```

```
Yes company to allow
```

[40]:
```
#QUESTION(6)
# ANSWER
class passingmark(Exception):

    def __init__(self,quali):
        self.quali=quali
```

[41]:
```
def passedmark(marks):
    if marks<0:
        raise passingmark("marked not sufficient for qualifed")
    elif marks>600:
        raise passingmark("mark is very very high")
    else:
        print("over all good")
```

[19]:
```
try:
    marks=int(input('enterd your mark'))
    passedmark(marks)

except passingmark as e:
    print(e)
```

```
enterd your mark 56
over all good
```