12 FEB ASS

March 1, 2023

[ ]:
```
Q1. What is an Exception in python? write the difference between Exception and␣
 ↪syntax errors.
```

[ ]:
```
An exception is an event, which occurs during the execution of a program that␣
 ↪disrupts the normal flow of the program
instructions. in general , when a python script encounters a situation that it␣
 ↪cannot cope with , it raises an exception.
an exception is a python object that represents an error.
```

[ ]:
```
Both errors and exceptions are type of runtime error , which means they occur␣
 ↪during the execution of a program. in simple words,
the error is a critical issue that a normal application should not catch ,␣
 ↪while an exception is a condition that a program
should catch.
```

[ ]:
```

```

[ ]:
```

```

[ ]:
```
Q2. What happens when an exception is not handeled? explain with an example.
```

ANS -

[ ]:
```
If an exception occurs during execution of the  try caluse , the exception may␣
 ↪be handled by an except clause.
if the exception is not handled by an except clause , the exception is␣
 ↪re-raised after the finally clause
has been executed.
```

[19]:
```
try:
    f = open("test.txt" , "r")
    f.write("this is my method")
finally:
    print("this will always execute")
```

```
this will always execute
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[19], line 2
      1 try:
----> 2     f = open("test.txt" , "r")
      3     f.write("this is my method")
      4 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: 'test.txt'
```

[ ]:

[ ]:

[ ]:

[ ]:
```
Q3. which python statements are used to catch and handle exceptions? explain␣
 ↪with an examples.
```

ANS -

[ ]:
```
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 response to any exceptions
in the preceding try clause.
```

[18]:
```
try:
    f = open("test.txt" , "r")
    f.write("this is my print")
except Exception as e :
    print("there is some issues in the code" , e)
```

```
there is some issues in the code [Errno 2] No such file or directory: 'test.txt'
```

[ ]:

[ ]:

[ ]:
```
Q4. Explain with an example:
    a. try and else
    b. finally
    c. raise
```

ANS -

[8]:
```
f  = open("test.txt" , "r")
f.write("this is my print")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[8], line 1
----> 1 f  = open("test.txt" , "r")
      2 f.write("this is my print")

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: 'test.txt'
```

[9]:
```
try:
    f = open("test.txt" , "r")
    f.write("this is my print")
except Exception as e :
    print("there is some issues in the code" , e)
else:
    f.close()
    print("this block will execute once try will execute itself without an␣
 ↪exception")
```

```
there is some issues in the code [Errno 2] No such file or directory: 'test.txt'
```

[ ]:

[10]:
```
try:
    f = open("test.txt" , "r")
    f.write("this is my print")
except Exception as e :
    print("there is some issues in the code" , e)
else:
    f.close()
    print("this block will execute once try will execute itself without an␣
 ↪exception")
finally:
    print(" I will execute always")
```

```
there is some issues in the code [Errno 2] No such file or directory: 'test.txt'
 I will execute always
```

[ ]:

[11]:
```
class validatage(Exception):
    def __init__(self , msg):
        self.msg = msg
```

[12]:
```
def validatage_age(age):
    if age < 0 :
        raise validatage("age should not be less than zero")
    elif age > 100 :
        raise validatage("age should not be gretter than hundred")
    else:
        print("age is valid")
```

[13]:
```
try:
    age = int(input("enter your age"))
    validatage_age(age)
except validatage as e :
        print(e)
```

```
enter your age 22
```

```
age is valid
```

[ ]:

[ ]:

[ ]:
```
Q5. what are custom exception in python ? why do we need custom exception ?␣
 ↪explain with an example.
```

ANS -

[ ]:
```
Custom exception are also classes. hence , you can add functionally to the␣
 ↪custom exception classes like :
   adding attributes and properties. adding methods
Built-in exception offer information about python related problems , and custom␣
 ↪exceptions will add information about project related
problems. That way, you can design your code in a way that combines python code␣
 ↪with the language of the project.
```

[6]:
```
class invalid_age(Exception):
    raise invalid_age("age should not be less than 18")
    pass
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[6], line 1
----> 1 class invalid_age(Exception):
      2     raise invalid_age("age should not be less than 18")
      3     pass

Cell In[6], line 2, in invalid_age()
      1 class invalid_age(Exception):
----> 2     raise invalid_age("age should not be less than 18")
      3     pass

NameError: name 'invalid_age' is not defined
```

[7]:
```
try:
    input_num = int(input("enter a number: "))
    if input_num < 18 :
        raise invalid_age("the age should not be less than 18")
    else:
        print("Eligible to vote")
except invalid_age as e :
    print(e)
```

```
enter a number:  18
Eligible to vote
```

[ ]:

[ ]:

[ ]:
```
Q6. create a custom exception class. use this class to handle an exception.
```

ANS -

[2]:
```
class validatage(Exception):
    def __init__(self , msg):
        self.msg = msg
```

[3]:
```
def validatage_age(age):
    if age < 0 :
        raise validatage("age should not be less than zero")
    elif age > 100 :
        raise validatage("age should not be gretter than hundred")
    else:
        print("age is valid")
```

[5]:
```
try:
    age = int(input("enter your age"))
    validatage_age(age)
except validatage as e :
        print(e)
```

```
enter your age 22
```

```
age is valid
```

[ ]:

[ ]: