UNIT-V

File Handling, Exception Handling and Python Libraries for robotics

Unit-V File Handling, Exception Handling and Python Libraries for Robotics	5a. Read and data data on file. 5b. Handle the given exception. 5c.Install Python libraries.	5.1 File Handling:- Opening file in different modes, accessing file content, reading and writing file, closing file, renaming file. 5.2 Exception Handling:- Introduction, try: except, statement, raise, user defined exception. 5.3 Python Libraries:-- opencv, pybotics,DART

- **File Handling in Python**

Python too supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, but like other concepts of Python, this concept here is also easy and short. Python treats file differently as text or binary and this is important. Each line of code includes a sequence of characters and they form text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun. Let's start with Reading and Writing files.

➢ Working of open() function

Before performing any operation on the file like read or write, first we have to open that file. For this, we should use Python's inbuilt function open()

But at the time of opening, we have to specify the mode, which represents the purpose of the opening file.

Syntax:

File_object = open("File_Name", "Access_Mode")

Where the following access mode is supported:

1. **r:** open an existing file for a read operation.
2. **w:** open an existing file for a write operation. If the file already contains some data then it will be overridden.
3. **a:** open an existing file for append operation. It won't override existing data.
4. **r+:** To read and write data into the file. The previous data in the file will not be deleted.

5. **w+:** To write and read data. It will override existing data.
6. **a+:** To append and read data from the file. It won't override existing data.

Steps to open file:

- Assume we have the following file, located in the file demofile.txt

demofile.txt Hello! Welcome to demofile.txt This file is for testing purposes. Good Luck!

- To open the file, use the built-in open() function.
- The open() function returns a file object, which has a read() method for reading the content of the file:

Example

```
f = open("demofile.txt", "r")
print(f.read())
```

Output:

➢ Reading from a file

There are three ways to read data from a text file.

- **read() :** Returns the read bytes in form of a string. Reads n bytes, if no n specified, reads the entire file.
- File_object.read([n])
- **readline() :** Reads a line of the file and returns in form of a string.For specified n, reads at most n bytes. However, does not reads more than one line, even if n exceeds the length of the line.
- File_object.readline([n])
- **readlines() :** Reads all the lines and return them as each line a string element in a list. File_object.readlines()

Example:

demofile.txt Hello! Welcome to demofile.txt This file is for testing purposes. Good Luck!

- **read() :**

```
file1 = open("C:/Users/Dell/Desktop/demofile.txt", "r+")
```

print("Output of Read function is: ")

print(file1.read())

Output:

Output of Read function is:

Hello

Welcome to demofile.txt

This file is for testing purposes.

Good Luck!

- **readline()**

print("Output of Readline function is ")

print(file1.readline(5))

Output:

Hello

- **readlines()**

print("Output of Readlines function is ")

print(file1.readlines())

Output:

['Hello\n']

➢ Closing a file in Python:

Python has a **close()** method to close a file. The close() method can be called more than once and if any operation is performed on a closed file it raises a ValueError. The below code shows a simple use of close() method to close an opened file.

Example1:

```
# open the file using open() function
>>> file1 = open("C:/Users/Dell/Desktop/demofile.txt")
# Reading from file
>>> print(file1.read())
# closing the file
file1.close()
```

Here file is opened and we rea it and closed the file.

Now if we try to perform any operation on a closed file like shown below it raises a ValueError:

Now if we try to perform any operation on a closed file like shown below it raises a ValueError:

```
>>> print(file1.read())
Traceback (most recent call last):
  File "<pyshell#23>", line 1, in <module>
    print(file1.read())
ValueError: I/O operation on closed file.
```

➢ Working of append() mode

Let's see how the append mode works:

```
# Python code to create a file
>>> file1 = open("C:/Users/Dell/Desktop/demofile.txt","a")
>>> file1.write("Now the file has more content!")
>>>file1.close()
```

Output:

```
file1 = open("C:/Users/Dell/Desktop/demofile.txt")
>>> print(file1.read())
Hello
Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
Now the file has more content!
```

The close() command terminates all the resources in use and frees the system of this particular program.

➢ Creating a file using write() mode

Let's see how to create a file and how write mode works:
To manipulate the file, write the following in your Python environment:

```
>>>file1 = open("C:/Users/Dell/Desktop/demofile.txt","w")
>>> file1.write("Woops! I have deleted the content!")
>>> file1.close()
```

Output:

```
>>> file1 = open("C:/Users/Dell/Desktop/demofile.txt","r")
>>> print(file1.read())
Woops! I have deleted the content!
```

the "w" method will overwrite the entire file.

➢ Renaminfg a file:

Steps to Rename a File using Python

Suppose that your goal is to rename a *text file* from "demofile.txt" to "newdemofile.txt"
Here are the steps that you may follow to rename your file:

Step 1: Capture the path where the file is stored

To start, capture the path where your file is stored.

For demonstration purposes, let's suppose that a file called "demofile" is stored under the following path:

C:/Users/Dell/Desktop

Note that you'll need to modify the file path to reflect the location where the file is stored on *your* computer.

Step 2: Rename the file

To rename the file using Python, you'll need to import the ***os*** **package**.
You can then use the following template to rename your file:

```
import os

os.rename(r'file path\OLD file name.file type',r'file path\NEW file name.file type')
```

In the context of our example:

- File path: C:\Users\Ron\Desktop\Test
- OLD file name: demofile
- NEW file name: newdemofile
- File type: txt

Don't forget to put "r" before the file path to avoid the following error in Python:

(unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: truncated \UXXXXXXXX escape

The complete Python code to rename the text file from "demofile" to "newdemofile" is:

```
import os

os.rename(r'C:/Users/Dell/Desktop/demofile.txt',r'C:/Users/Dell/Desktop/newdemofile.txt')
```

Run the code (adjusted to your file path) and you'll get the new file name:

newdemofile

• Python Exception Handling

Error in Python can be of two types i.e. Syntax errors and Exceptions. Errors are the problems in a program due to which the program will stop the execution. On the other hand, exceptions are raised when some internal events occur which changes the normal flow of the program.

Difference between Syntax Error and Exceptions

Syntax Error: As the name suggests this error is caused by the wrong syntax in the code. It leads to the termination of the program.

Example:

```
amount = 10000

# check that You are eligible to

if(amount > 2999)

print("You are eligible to purchase ")
```

Output:

```
  File "/home/ac35380186f4ca7978956ff46697139b.py", line 4
    if(amount>2999)
                  ^
SyntaxError: invalid syntax
```

Exceptions: Exceptions are raised when the program is syntactically correct, but the code resulted in an error. This error does not stop the execution of the program, however, it changes the normal flow of the program.

Example:

```
# initialize the amount variable

marks = 10000


# perform division with 0

a = marks / 0

print(a)
```

Output:

```
Traceback (most recent call last):
  File "/home/f3ad05420ab851d4bd106ffb04229907.py", line 4, in <module>
    a=marks/0
ZeroDivisionError: division by zero
```

In the above example raised the ZeroDivisionError as we are trying to divide a number by 0.

Note: Exception is the base class for all the exceptions in Python.

➢ Try and Except Statement – Catching Exceptions

Try and except statements are used to catch and handle exceptions in Python. Statements that can raise exceptions are kept inside the try clause and the statements that handle the exception are written inside except clause.

Example: Let us try to access the array element whose index is out of bound and handle the corresponding exception.

```
# Python program to handle simple runtime error

#Python 3

a = [1, 2, 3]
try:
    print ("Second element = %d" %(a[1]))

    # Throws error since there are only 3 elements in array
    print ("Fourth element = %d" %(a[3]))

except:
    print ("An error occurred")
```

Output

Second element = 2

An error occurred

In the above example, the statements that can cause the error are placed inside the try statement (second print statement in our case). The second print statement tries to access the fourth element of the list which is not there and this throws an exception. This exception is then caught by the except statement.

➢ Catching Specific Exception

A try statement can have more than one except clause, to specify handlers for different exceptions. Please note that at most one handler will be executed. For example, we can add Index Error in the above code.

The general syntax for adding specific exceptions are –

```
try:
    # statement(s)
except IndexError:
```

```
    # statement(s)
except ValueError:
    # statement(s)
```

Example: Catching specific exception in Python

```
# Program to handle multiple errors with one
# except statement
# Python 3

def fun(a):
    if a < 4:

        # throws ZeroDivisionError for a = 3
        b = a/(a-3)

    # throws NameError if a >= 4
    print("Value of b = ", b)

try:
    fun(3)
    fun(5)

# note that braces () are necessary here for
# multiple exceptions
except ZeroDivisionError:
    print("ZeroDivisionError Occurred and Handled")
except NameError:
    print("NameError Occurred and Handled")
```

Output

ZeroDivisionError Occurred and Handled

If you comment on the line fun(3), the output will be

NameError Occurred and Handled

The output above is so because as soon as python tries to access the value of b, NameError occurs.

➢ Raising Exception

The raise statement allows the programmer to force a specific exception to occur. The sole argument in raise indicates the exception to be raised. This must be either an exception instance or an exception class (a class that derives from Exception).

```
# Program to depict Raising Exception
```

```
try:
    raise NameError("Hi there")  # Raise Error
except NameError:
    print ("An exception")
    raise  # To determine whether the exception was raised or not
```

The output of the above code will simply line printed as “An exception” but a Runtime error will also occur in the last due to the raise statement in the last line. So, the output on your command line will look like

```
An exception
Traceback (most recent call last):
File "C:/Users/Dell/1..py", line 2, in <module>
raise NameError("Hi there")  # Raise Error
NameError: Hi there
```

- **Python libraries:**
  - **Opencv:**

OpenCV is a great tool for **image processing and performing computer vision tasks**. It is an open-source library that can be used to perform tasks like face detection, objection tracking, landmark detection, and much more. It supports multiple languages including python, java C++

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today’s systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it’s free for both **academic** and **commercial** use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing.

Applications of OpenCV:

There are lots of applications which are solved using OpenCV, some of them are listed below

- face recognition

- Automated inspection and surveillance
- number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anamoly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition

➢ Pybotics:

It is an open-source Python toolbox for robot kinematics and calibration. It was designed to provide a simple, clear, and concise interface to quickly simulate and evaluate common robot concepts, such as kinematics, dynamics, trajectory generations, and calibration. The toolbox is specifically designed for use with the Modified Denavit–Hartenberg parameters convention.

Installation

```
# python3 is mapped to pip or inside a venv
pip install pybotics
```

Applications of Pybotics

- Basic Usage
- Kinematics
- Calibration
- Trajectory and Path Planning
- Machine Learning
- Dynamics

➢ DART Library:

Dart is the open-source programming language originally developed by Google. It is meant for both the server-side and the user side. The Dart SDK comes with its compiler – the **Dart VM** and a utility dart2js which is meant for generating **JavaScript equivalent of a Dart Script** so that it can be run on those sites also which don't support Dart.
Dart is an **Object-oriented language** and is quite similar to that of **Java Programming**. Dart is extensively used to create single-page websites and web-applications. The best **example** of a dart application is **Gmail**.