Python Libraries, Exception Handling

Distributed Network and Systems Laboratory,
Chonnam National University
Presenter : Shivani Kolekar

Recap

- ✓ **Introduction to Python libraries**
- ✓ **math, random etc.**
- ✓ **Installing libraries – pandas etc.**

Today's Goals

- ✓ **Introduction to Errors**
- ✓ **Exceptions**
- ✓ **types**
- ✓ **Try-Except Statement**
- ✓ **Finally keyword**
- ✓ **Q & A**

Modules

- A module is a Python source code file that contains functions and/or classes.
- Ex. – math, random
- To import a module, you write an import statement at the top of your program.
- General Format with example of math module :

import math

This statement causes the Python interpreter to load the contents of the math module into memory, making the functions and/or classes that are stored in the math module available to the program.

Random module

- Python has a built-in module that you can use to make random numbers.
- The random module has a set of methods:

randint()	Returns a random number between the given range
choice()	Returns a random element from the given sequence

- General Format with example of math module :

import random

Random module

- Python has a built-in module that you can use to make random numbers.

random_number.py

```
import random

# Generate a random number between 1 and 10
random_number = random.randint(1, 10)
print("Random Number:", random_number)
```

Output

```
Random Number: 2
```

math module

- ✓ importing math module from Python:

Square_root1.py

```
import math

x = math.sqrt(25)
print(x)
```

```
5.0
```

Output

- ✓ Using alias for importing math:

import math as mt

Using_alias.py

```
import math as mt
x = mt.sqrt(25)
a = mt.sqrt(1024)
print("square root of 25: \n", x)
print("square root of 1024: \n", a)
```

```
square root of 25:
 5.0
square root of 1024:
 32.0
```

Output

Using pandas

- Pandas is a Python library used for working with data sets.
- It has functions for analyzing, cleaning, exploring, and manipulating data.

Pandas_example.py

```
import pandas as pd

file_path = 'example_data.xlsx'

df = pd.read_excel(file_path)
print(df)
```

Output

```
                                          Food Item Measure  Calories  \
0   Carrot, homemade with cream cheese icing (2 la...    1/12       542
1            Cereal bar, fruit filled (Nutri-GrainTM)       1       135
2                 Cheesecake, commercial (15 cm diam)     1/6       321
3     Cheesecake, from mix, no-bake type (20cm diam)     1/8       407
4   Cheesecake, plain, homemade with cherry toppin...     1/8       459
5              Cherry, commercial, 2 crust (23cm diam)     1/8       325
```

Errors vs Exceptions

- ✓ errors are situations a program can't do anything about, like a disk failure or a RAM failure.
- ✓ Exceptions are situations a program can overcome, like say you try to open a file and it doesn't exist.

Different types of errors in Python

- ✓ SyntaxError
- ✓ TypeError
- ✓ IndexError
- ✓ ValueError
- ✓ ZeroDivisionError
- ✓ ImportError

Syntax error

- ✓ This error is caused by the wrong syntax in the code. It leads to the termination of the program.
- ✓ Head to the colab notebook for exercise example!

example1.py

What could be the output ?

```
amount = 10000
if(amount > 2999) :
        print("You are eligible to purchase the game")
```

Exceptions in Python

✓ **Exceptions**:

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

```
marks = 100
a = marks / 0
print(a)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-16-cdaef1118976> in <cell line: 2>()
      1 marks = 100
----> 2 a = marks / 0
      3 print(a)

ZeroDivisionError: division by zero
```

Try and Except Statement

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

```
try:
   statement
   statement
   etc.
except   ExceptionName:
   statement
   statement
   etc.
```

ZeroDivisionError

✓ This exception is raised when an attempt is made to divide a number by zero.

✓ It can be handled with try and except statement.

Division.py

```
marks = 100
a = marks / 0
print(a)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call
<ipython-input-6-cdaef1118976> in <cell line: 2>()
      1 marks = 100
----> 2 a = marks / 0
      3 print(a)

ZeroDivisionError: division by zero
```

Division2.py

```
marks = 100
try:
    a = marks / 0

except ZeroDivisionError:
    print('Cannot divide by zero')
```

```
Cannot divide by zero
```

ValueError

- ✓ This exception is raised when a function or method is called with an invalid argument or input, such as
- ✓ trying to convert a string to an integer when the string does not represent a valid integer.

Input.py

```
# Determining the grade level from user input
user_input = input("Enter your grade (9-12): ")

try:
    grade = int(user_input)
    if 9 <= grade <= 12:
        print(f"You are in grade {grade}.")
    else:
        print("Grade must be between 9 and 12.")
except ValueError:
    print("Error: Please enter a numeric grade level.")
```

```
Enter your grade (9-12): Third
Error: Please enter a numeric grade level.
```

TypeError

✓ This exception is raised when an operation or function is applied to an object of the wrong type.

- How will you handle this error?

addition.py

```
x = 5
y = "hello"
z = x + y
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-1-01628386ab26> in <cell line: 3>()
      1 x = 5
      2 y = "hello"
----> 3 z = x + y

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

IOError

- ✓ This exception is raised when an I/O operation, such as reading or writing a file, fails due to an input/output error.

File.py

```
def main():
    filename = input('Enter a filename: ')
    infile = open(filename, 'r')

    contents = infile.read()
    print(contents)

    infile.close()
main()
```

```
Enter a filename: bad_file.txt
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last
<ipython-input-20-1af089587a72> in <cell line: 19>()
     17
     18 # Call the main function.
---> 19 main()

<ipython-input-20-1af089587a72> in main()
      5
      6     # Open the file.
----> 7     infile = open(filename, 'r')
      8
      9     # Read the file's contents.

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

perform exception handling →

File_try_except.py

```
def main():
    # Get the name of a file.
    filename = input('Enter a filename: ')

    try:
        # Open the file.
        infile = open(filename, 'r')

        # Read the file's contents.
        contents = infile.read()

        # Display the file's contents.
        print(contents)

        # Close the file.
        infile.close()
    except IOError:
        print('An error occurred trying to read')
        print('the file', filename)

# Call the main function.
main()
```

Output

```
Enter a filename: bad_file.txt
An error occurred trying to read
the file bad_file.txt
```

Finally Keyword

- ✓ Python provides a keyword finally, which is always executed after the try and except blocks.
- ✓ The finally block always executes after the normal termination of the try block or after the try block terminates due to some exception.

```
try:
    # Some Code....

except:
   # Handling of exception

finally:
    # Some code..... (always executed)
```

Lets Code!