MODULE IV

Python Modules - Importing Modules, Math Module, Random Module, Packages, Compositions.

Python file handling: Reading files, writing files, loading data, working with and saving data.

Exception and Error Handling, Regular Expression, Enumerate, High Order Functions Lambda, Filter, Map, Reduce

RA20 APTT

Modules vs Packages

Module: are simply files with the **". py"** extension containing Python code that can be imported inside another Python Program. Module contains functions and global variables . It is an executable file and to organize all the modules we have the concept called Package in Python. E.g from math import pow, import math

Packages : A python package is a collection of modules. Modules that are related to each other are mainly put in the same package. When a module from an external package is required in a program, that package can be imported and its modules can be put to use. This directory contains Python modules also have __init__.py file by which the interpreter interprets it as a Package.

RA20 APTT

Math Module

•The **math module** is a standard module in Python and is always available.

•Python math module contains most famous mathematical functions, which includes trigonometric functions, representation functions, logarithmic functions, etc.

• Math module also defines two mathematical constants, i.e.Pie and Euler number.

•To use mathematical functions under this module, you have to import the module using import math.

import math or **import math as m**

math.sqrt(4) **m.sqrt(4)**

•Some funtions math.log10(20), math.pow(2,4), math.floor(10.4) , math.ceil(10.5), math.factorial() , math.modf(10.5)

RA20 APTT

List creation

Function	Description
ceil(x)	Returns the smallest integer greater than or equal to x.
copysign(x, y)	Returns x with the sign of y
fabs(x)	Returns the absolute value of x
factorial(x)	Returns the factorial of x
floor(x)	Returns the largest integer less than or equal to x
fmod(x, y)	Returns the remainder when x is divided by y
frexp(x)	Returns the mantissa and exponent of x as the pair (m, e)
fsum(iterable)	Returns an accurate floating point sum of values in the iterable
isfinite(x)	Returns True if x is neither an infinity nor a NaN (Not a Number)
isinf(x)	Returns True if x is a positive or negative infinity
isnan(x)	Returns True if x is a NaN
ldexp(x, i)	Returns x * (2**i)
modf(x)	Returns the fractional and integer parts of x
trunc(x)	Returns the truncated integer value of x
exp(x)	Returns e**x
expm1(x)	Returns e**x - 1
log(x[, b])	Returns the logarithm of x to the base b (defaults to e)
log1p(x)	Returns the natural logarithm of 1+x
log2(x)	Returns the base-2 logarithm of x
log10(x)	Returns the base-10 logarithm of x
pow(x, y)	Returns x raised to the power y

RA20 APTT

List creation

Function	Description
sqrt(x)	Returns the square root of x
acos(x)	Returns the arc cosine of x
asin(x)	Returns the arc sine of x
atan(x)	Returns the arc tangent of x
atan2(y, x)	Returns atan(y / x)
cos(x)	Returns the cosine of x
hypot(x, y)	Returns the Euclidean norm, sqrt(x*x + y*y)
sin(x)	Returns the sine of x
tan(x)	Returns the tangent of x
degrees(x)	Converts angle x from radians to degrees
radians(x)	Converts angle x from degrees to radians
acosh(x)	Returns the inverse hyperbolic cosine of x
asinh(x)	Returns the inverse hyperbolic sine of x
atanh(x)	Returns the inverse hyperbolic tangent of x
cosh(x)	Returns the hyperbolic cosine of x
sinh(x)	Returns the hyperbolic cosine of x
tanh(x)	Returns the hyperbolic tangent of x
pi	Mathematical constant, the ratio of circumference of a circle to it's diameter (3.14159...)
e	mathematical constant e (2.71828...)

RA20 APTT

Random Module

• Python offers **random module** that can generate **random numbers**.

• These are **pseudo-random number** & sequence of number generated depends on the seed.

•Python random module implements pseudo-random number generators for various distributions, including integer and float (real).

```
import random
print(random.random())
# Output 0.24480512307264823
```

You may get a different number

•The **random.random()** is the most basic function of the random module.

•Almost all functions of the random module depend on the basic function **random().**

•random() return the next random floating-point number in the range [0.0, 1.0].

RA20 APTT

Random

```python
import random

# random number from 0 to 1
print(random.random())
# Output 0.16123124494385477

# random number from 10 to 20
print(random.randint(10, 20))
# Output 18

# random number from 10 to 20 with step 2
print(random.randrange(10, 20, 2))
# Output 14

# random float number within a range
print(random.uniform(5.5, 25.5))
# Output 5.86390810771935

# random choice from sequence
print(random.choice([10, 20, 30, 40, 50]))
# Output 30
```

```python
# random sample from sequence
print(random.sample([10, 20, 30, 40, 50], k=3))
# Output [50, 10, 20]

# random sample without replacement
print(random.choices([10, 20, 30, 40, 50], k=3))
# Output [30, 10, 40]

# random shuffle
x = [10, 20, 30, 40, 50, 60]
random.shuffle(x)
print(x)
# [60, 10, 30, 20, 50, 40]

# random seed
random.seed(2)
print(random.randint(10, 20))
# 10
random.seed(2)
print(random.randint(10, 20))
# 10
```

RandInt and Randrange

- **randint()** Generate random integer number from the inclusive range. E.g random.randint(0, 10) will return random number from **[0, 1, 2, 3, 4, 5, 6, 7, 8 ,9, 10].**

- **randint** considers both the start and stop numbers while generating random integers

- **randrnage()** is used to generate random integer from the given **exclusive range** by specifying the increment. For example, **random.randrange(0, 10, 2)** will return any random number between 0 and 10 (like 0, 2, 4, 6, 8).

- The randrange() **doesn't consider the stop number** while generating a random integer. It is an exclusive random range.

- In **rangrange** Out of three, **two parameters are optional**. i.e., start and step

RA20 APTT

Random number of a specific length

```python
import random
# random number of length 4
num1 = random.randint(1000, 9999)

# random number of length 4 with step 2
num2 = random.randrange(1000, 10000, 2)
print(num1, num2)

# Output 3457 5116
```

RA20 APTT

Random negative integer

```python
import random

singed_int = random.randrange(-60, -6)
print(singed_int)
# Output -16
```

Random positive or negative integer

```python
import random

for i in range(5):
    print(random.randint(-10, 10), end=' ')
# Output 10 -1 5 -10 -7
```

RA20 APTT

List of Random Integers

```python
import random

random_list = []
# Set a length of the list to 10
for i in range(0, 10):
    # any random numbers from 0 to 1000
    random_list.append(random.randint(0, 1000))
print(random_list)
# Output [994, 287, 65, 994, 936, 462, 839, 160, 689, 624]
```

RA20 APTT

List of Random Integers Without Duplicates

- To make sure each number in the list is unique, use the **random.sample()** method to generate a **list of unique random numbers.**

- The **sample() returns** a sampled list of selected random numbers within a range .

- **Sample () never repeats the element** so we get a list of random numbers **without duplicates.**

```python
import random

# Generate 10 unique random numbers within a range
num_list = random.sample(range(0, 1000), 10)
print(num_list)
# Output [499, 580, 735, 784, 574, 511, 704, 637, 472, 211]
```

RA20 APTT

Generate a secure random integer

- Till now all examples of random numbers are not cryptographically secure.
- The **cryptographically secure random generator** generates random numbers using **synchronization methods** to ensure that no two processes can obtain the same number simultaneously.
- For a security-sensitive application use **secret module** (Available in Python version higher than 3.6.)

```python
import secrets

# secure random integer
# from 0 to 10
secure_num = secrets.randbelow(10)
print(secure_num)
# Output 5
```

RA20 APTT

Summaryabout randint() and randrange()

- **Use randint()** when you want to generate a random number from an inclusive range.
- Use **randrange()** when you want to generate a random number from an exclusive range by specifying the increment.
- The randint() rand randrange() works only with integers not with float numbers.
- In randrange() step must not be 0, or else you will get a ValueError.
- The start should not be greater than stop if you are using all positive numbers.
- **If start is greater than stop**, you will get a **ValueError in** randrange(), But, you **can set a start value greater than stop** if you are using a **negative step value**.

```python
import random

# ValueError: empty range for randrange()
print(random.randrange(100, 10, 2))
```

```python
import random

print(random.randrange(100, 10, -2))
# output 60
```

Random String in Python

- To generate **random strings** in Python, the **string and random modules** are used.
- random.choice() is used to generate strings in **which characters may repeat**, while . **random.sample() is used for non-repeating characters**

Method	Description
string.ascii_uppercase	Returns a string with uppercase characters
string.ascii_lowercase	Returns a string with lowercase characters
string.ascii_letters	Returns a string with both lowercase and uppercase characters
string.digits	Returns a string with numeric characters
string.punctuation	Returns a string with punctuation characters

RA20 APTT

Random String in Python

```python
import random
import string

# printing lowercase
letters = string.ascii_lowercase
print ( ''.join(random.choice(letters) for i in range(10)) )

# printing uppercase
letters = string.ascii_uppercase
print ( ''.join(random.choice(letters) for i in range(10)) )

# printing letters
letters = string.ascii_letters
print ( ''.join(random.choice(letters) for i in range(10)) )

# printing digits
letters = string.digits
print ( ''.join(random.choice(letters) for i in range(10)) )

# printing punctuation
letters = string.punctuation
print ( ''.join(random.choice(letters) for i in range(10)) )
```

RA20 APTT

Random String in Python

```python
import random
import string
# printing lowercase
letters = string.ascii_lowercase
print_(_''.join(random.choice(letters) for i in range(10))_)
```

```python
import random
import string

def get_rand_string(length):
    letters = string.ascii_lowercase
    print_(_''.join(random.choice(letters) for i in range(length)))

get_rand_string(8)
get_rand_string(6)
get_rand_string(4)
```

Random String of mix Case and Digits

```python
import random
import string

def get_rand_string(length):
    letters = string.ascii_letters
    print_(_''.join(random.choice(letters) for i in range(length)))

get_rand_string(8)
get_rand_string(6)
get_rand_string(4)
```

```python
import random
import string
def get_rand_string(length):
    letters = string.digits
    print_(_''.join(random.choice(letters) for i in range(length)))
get_rand_string(8)
get_rand_string(6)
get_rand_string(4)
```

Random String of Specific letters & Without Repeating Characters

```
import random
import string

print(''.join(random.choice("abcdxyzpqr") for i in range(5)))
```

```
import random
import string

for i in range(3):
    # get random string of length 6 without repeating letters
    result_str = ''.join(random.sample(string.ascii_lowercase, 5))
    print(result_str)
```

RA20 APTT

Create Password

```
import random
import string

# get random password pf length 8 with letters, digits, and symbols
mixed_chars = string.ascii_letters + string.digits + string.punctuation
password = ''.join(random.choice(mixed_chars) for i in range(8))
print("Random password is:", password)
```

```
import random
import string

password = ''.join(random.choice(string.printable) for i in range(8))
print("Random password is:", password)
```

import string

Storing the sets of punctuation, digits, ascii_letters and spaces in variable result . It's a Pre-initialized string used as string constant

result = string.printable
print(result)

RA20 APTT

Practice Questions

1. Produce 10 random integers between 100 and 999 .
2. Produce 20 random lottery tickets and pick two lucky tickets as a winner. The lottery number must be 10 digits long All 20 ticket number must be unique.
3. Generate 6 digit random secure OTP
4. Choose a random character from a given String.
5. Generate a random Password which is 10 characters long and have at least 1 digit, and 1 special symbol.
6. Dice is Rolled in such a way that every time we get the same number
7. Multiplication of two random float numbers num1 between 0.1 and 1, num2 between 9.5 and 99.5

RA20 APTT

Practice Questions

Q 1

```
import random
print("Generating 10 random integer number between 100 and 999)
for num in range(10):
    print(random.randrange(100, 999   ), end=', ')
```

Q 2

```
import random
lot_tickets_list = []
print("creating 20 random lottery tickets")
# to get 100 ticket
for i in range(20):
    lot_tickets_list.append(random.randrange(1000000000, 9999999999))
# pick 2 luck tickets
winners = random.sample(lot_tickets_list, 2)
print("Lucky 2 lottery tickets are", winners)
```

RA20 APTT

Practice Questions

Q 3

```
import secrets
#Getting systemRandom class instance out of secrets module
sec_generator = secrets.SystemRandom()

otp = sec_generator.randrange(1000, 9999)
print("Secure Random OTP = ", otp)
```

Q 4

```
import random

name = 'Welcome to Python'
char = random.choice(name)
print("random char is ", char)
```

RA20 APTT

Practice Questions

Q 5

```
import random
import string

def randomPassword():
    randomSource = string.ascii_letters + string.digits + string.punctuation
    password = random.sample(randomSource, 6)
    password += random.sample(string.ascii_uppercase, 2)
    password += random.choice(string.digits)
    password += random.choice(string.punctuation)

    passwordList = list(password)
    random.SystemRandom().shuffle(passwordList)
    password = ''.join(passwordList)
    return password

print ("Password is ", randomPassword())
```

RA20 APTT

Practice Questions

Q 6

```python
import random

dice = [1, 2, 3, 4, 5, 6]
print("Randomly selecting same number of a dice")
for i in range(5):
    random.seed(25)
    print(random.choice(dice))
```

Q 6

```python
import random

print('Random number with seed 30')
for i in range(3):
    # Random number with seed 30
    random.seed(30)
    print(random.randint(25, 50))
```

Q 7

```python
import random

num1  = random.random()
print("First Random float is ", num1)
num2 = random.uniform(9.5, 99.5)
print("Second Random float is ", num1)

num3 = num1 * num2
print("Multiplication is ", num3)
```

RA20 APTT

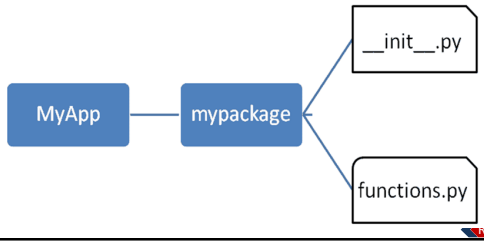
PACKAGES

- A package in Python takes the concept of the modular approach.
- A package can contain one or more relevant modules.
- A module can contain multiple objects, such as classes, functions, etc.
- Let us create a package named mypackage, using the following steps:
  - Create a new folder named e.g. C:\myapp
  - Inside myapp, create a subfolder with the name 'mypackage'.
  - Create an empty __init__.py file in the mypackage folder.
  - Using a Python-aware editor like Pycharm, IDLE, create modules greet.py and functions.py

RA20 APTT

PACKAGES

- A package in Python takes the concept of the modular approach.
- A package can contain one or more relevant modules.
- A module can contain multiple objects, such as classes, functions, etc.



RA20 APTT

FILE HANDLING

- Files are **named locations** on disk to store related information.
- They are used to **permanently store data** in a non-volatile memory.
- Since Random Access Memory (RAM) is volatile, so files are used for future use of the data by permanently storing them.
- When we want to read from or write to a file, we need to open it first.
- When we are done, it needs to be closed so that the resources that are tied with the file are freed.
- In Python, a **file operation t**akes place in the following order:
  - **Open a file**
  - **Read or write a file**
  - **Close the file**

RA20 APTT

FILE Type and File Paths

Types

- **Text File**: we normally store character data, e.g. **test.txt.**
- **Binary File**: The binary files are used to store binary data such as images, video files, audio files, etc.

File Path : A file path defines the location of a file or folder in the computer system. There are two ways to specify a file path.

- **Absolute path**: which always begins with the root folder.
- **Relative path**: which is relative to the program's current working directory.

RA20 APTT

READ FILE

- We don't have to import any module to create a new file .
- To **read or write a file**, we need to **open** that file.
- Python provides a **built-in function open().**
- Pass file path and access mode to the open

  file_object = open (filename / Path , access_mode)

- **Open** returns the **file object** that is used to read or write the file according to the access mode.
- Access mode represents the purpose of opening the file, e.g, R is for reading and W is for writing.

RA20 APTT

FILE OPERATIONS

•The open() function takes two parameters; **filename, and mode and main modes of operation** on a file:

w	It opens file to write mode only. It **overwrites the file if it exists** or creates a new one if no file exists with the same name. The file pointer exists at the beginning of the file
x	it creates a new file with the specified name. It causes an error a file exists with the same name.
a	It opens the file in the append mode. The file pointer exists at the end of the previously written file if exists any. It **creates a new file if no file exists with the same name**
R	The file pointer exists at the beginning. File by default is in open mode if no access mode is passed
r+	It opens the file to read and write both. The file pointer exists at the beginning of the file.
w+	It opens the file to write and read both. The file pointer exists at the beginning of the file
a+	It opens a file to append and read both. The file pointer remains at the end of the file if a file exists. It creates a new file if no file exists with the same name.

RA20 APTT

Creating a Text File

. X causes an error , if a file exists with the same name.

```
fp = open("Test.txt")    # open file in current directory
fp = open("C:/Python/Test.txt")  # specifying full path
```

```
# create a empty text file
# in current directory
fp = open('Test.txt', 'x')
fp.close()

# create a empty text file
fp = open('Test.txt', 'w')
fp.write('first line')
fp.close()
```

RA20 APTT

Reading a Text File

```
fp = open("Test.txt", "r")   # File Pointer
print(fp.read())   # Reading full file
print(fp.read(15))  # Reading only 15 characters from file
print(fp.readline())  # Reading a line form a file

for i in fp:   # Reading  full file line by line
    print(i)

fp.close()
```

RA20 APTT

Writing or Appending a Text File

```
fp = open("Test.txt", "a")   # File Pointer
fp.write("Now the file has more content!") # Appending Text
fp.close()


#open and read the file after the appending:
fp = open("Test.txt", "r")
print(fp.read())
```

RA20 APTT

Input Text From User

```
INPUT = input("Enter Your Text: \t")
fp = open("Test.txt", "w")
fp.write(INPUT)
fp = open("Test.txt", "r")
print(fp.read())
```

RA20 APTT

File Methods

Method	Description
read()	Returns the file content.
readline()	Read single line
readlines()	Read file into a list
truncate(size)	Resizes the file to a specified size.
write()	Writes the specified string to the file.
writelines()	Writes a list of strings to the file.
close()	Closes the opened file.
seek()	Set file pointer position in a file
tell()	Returns the current file location.
fileno()	Returns a number that represents the stream, from the operating system's perspective.
flush()	Flushes the internal buffer.

RA20 APTT

File Methods

•The seek() method is used to change or move the file's handle position to the specified location. The cursor defines where the data has to be read or written in the file.

•The tell() method return the **current position of the file pointer** from the beginning of the file.

```
fp = open("Test.txt", "r")
# move to 8 character
fp.seek(8)
# read from 11th character
print(fp.read())
print(fp.tell())
```

RA20 APTT

File Methods

```
fp = open("Test.txt", "r+")
str = fp.read(10)
print("Read String is : ", str)

# Check current position
position = fp.tell()
print("Current file position : ", position)

# Reposition pointer at the beginning once again
position = fp.seek(0, 0);
str = fp.read(10)
print("Again read String is : ", str)
# Close opend file
fp.close()
```

```
import os

# Rename a file from test1.txt to test2.txt
os.rename("Test.txt", "Renamed.txt")
```

```
import os

# Delete file Test2.txt
os.remove("Renamed.txt")
```

RA20 APTT

Load Data , Reading & Writing CSV files

While the open() method can read and write to both .txt and .csv files,

```
with open("Test.txt") as fp:
    for line in fp:
        #do something with line
        print(line)
```

When using the **CSV library,** we use **open() function** to open the file, and CSV reader() or writer() methods to **read from or write to a file.**

```
import csv

with open('students.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["Name", "Roll_No", "Address", "Phone"])
    writer.writerow(["Vikram", 1001, "Warangal", "2055566656"])
```

Reader Function

```
import csv

with open('students.csv') as file:
    reader = csv.reader(file, delimiter=',')
    for row in reader:
        print(row)
```

```
import csv

header = ["Name", "Roll_No", "Address", "Phone"]
data = [
        ['Vikram', 1001, 'Warangal', '2055566656'],
        ['Manikanta', 1002, 'Hostel', '2055567777'],
        ['Bhavani', 1003, 'Ananthasagar', '8066335577'],
        ['Ananya', 1003, 'Hanamkonda', '9995552525'],
        ['Bharat', 1004, 'KU', '6068779988'],
    ]

with open('students.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(header)
    writer.writerows(data)
```

writer.writerows(data)

APTT

Load Data With Built-In Functions

```
import csv

with open('airquality.csv') as fp:
    reader = csv.DictReader(fp, delimiter=',')
    for row in reader:
        print('Ozone is ' + row['Ozone'] + " With Solar Rad " +
              row['Solar.R'] + " and Wind is Blowing at " +
              row['Wind'] + 'With Temp' + row['Temp'])
```

Air quality is another **csv file**
You can use students file

```
Unit2
Ozone is 12 With Solar Rad 149 and Wind is Blowing at 12.6With Temp74
Ozone is 18 With Solar Rad 313 and Wind is Blowing at 11.5With Temp62
Ozone is 23 With Solar Rad 299 and Wind is Blowing at 8.6With Temp65
Ozone is 19 With Solar Rad 99 and Wind is Blowing at 13.8With Temp59
Ozone is 8 With Solar Rad 19 and Wind is Blowing at 20.1With Temp61
Ozone is 16 With Solar Rad 256 and Wind is Blowing at 9.7With Temp69
Ozone is 11 With Solar Rad 290 and Wind is Blowing at 9.2With Temp66
Ozone is 14 With Solar Rad 274 and Wind is Blowing at 10.9With Temp68
```

RA20 APTT

Load Data With Built-In Functions

```
with open('airquality.csv') as fp:
    reader = csv.DictReader(fp, delimiter=',')
    for row in reader:
        print(dict(row))
```

```
with open('airquality.csv') as ... > for row in reader
Unit2
{'Ozone': '12', 'Solar.R': '149', 'Wind': '12.6', 'Temp': '74', 'Month': '5', 'Day': '3'}
{'Ozone': '18', 'Solar.R': '313', 'Wind': '11.5', 'Temp': '62', 'Month': '5', 'Day': '4'}
{'Ozone': '23', 'Solar.R': '299', 'Wind': '8.6', 'Temp': '65', 'Month': '5', 'Day': '7'}
{'Ozone': '19', 'Solar.R': '99', 'Wind': '13.8', 'Temp': '59', 'Month': '5', 'Day': '8'}
{'Ozone': '8', 'Solar.R': '19', 'Wind': '20.1', 'Temp': '61', 'Month': '5', 'Day': '9'}
{'Ozone': '16', 'Solar.R': '256', 'Wind': '9.7', 'Temp': '69', 'Month': '5', 'Day': '12'}
{'Ozone': '11', 'Solar.R': '290', 'Wind': '9.2', 'Temp': '66', 'Month': '5', 'Day': '13'}
{'Ozone': '14', 'Solar.R': '274', 'Wind': '10.9', 'Temp': '68', 'Month': '5', 'Day': '14'}
{'Ozone': '18', 'Solar.R': '65', 'Wind': '13.2', 'Temp': '58', 'Month': '5', 'Day': '15'}
```

RA20 APTT

Reader Function

While using WITH release all resources at the end without closing the file

```
import csv

fp = open('students.csv')
reader = csv.reader(fp)
header = next(reader)
print(header)
```

```
import csv

with open('students.csv') as fp:
    reader = csv.reader(fp)
    header = next(reader)
print(header)
```

RA20 APTT

Reader Function

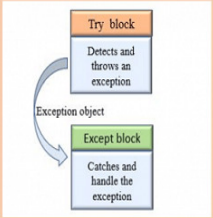
```
import csv

header = ["Name", "Roll_No", "Address", "Phone"]
data = [
        {'Name':'Vikram', 'Roll_No':1001 ,'Address': 'Warangal','Phone': 2055566656},
        {'Name':'Manikanta', 'Roll_No':1002 , 'Address':'Hostel', 'Phone':2055567777},
        {'Name':'Bhavani', 'Roll_No':1003 , 'Address':'Ananthasagar', 'Phone':8066335577},
        {'Name':'Ananya', 'Roll_No':1003 , 'Address':'Hanamkonda', 'Phone':9995552525},
        {'Name':'Bharat', 'Roll_No':1004 , 'Address':'KU', 'Phone':6068779988},
    ]

with open('std.csv', 'w', newline='') as file:
        writer = csv.DictWriter(file, fieldnames = header)
        writer.writeheader()
        writer.writerows(data)
```

Exception and Error Handling

Python Exception Handling

An exception is an event that occurs during the execution of programs that **disrupt the normal flow of execution**

1.Try-except-finally
2.Raise an exception
3.Exception chaining
4.Built-in exceptions
5.Custom exceptions



RA20 APTT

Exception and Error Handling

•An exception is an object derives from the **BaseException** class that contains information about an error event that occurred within a method.

•Exception object contains:

- ✓ Error type (exception name), S
- ✓ State of the program when the error occurred
- ✓ An error message describes the error event.

•Exception are useful to indicate different types of possible failure condition. e.g

1. FileNotFoundException
2. ImportError
3. RuntimeError
4. NameError
5. TypeError

RA20 APTT

Why Use Exception

Standardized error handling: Using built-in or creating a custom exception with precise name and description, we adequately define error event, that helps debug the error event.

Cleaner code: Exceptions separate the error-handling code from regular code, which helps us to maintain large code easily.

Robust application: An application is developed, which can handle error event efficiently

Exceptions propagation: Exception propagates example, if any error event occurred in a nested function, you do not have to explicitly catch-and-forward it; automatically, it gets forwarded to the calling function where you can handle it.

Different error types: We can use group errors by their generalized parent class, or Differentiate errors by their actual class.

RA20 APTT

Errors Types

•Syntax errors : The syntax error occurs when we are not following the proper structure or syntax of the language. A syntax error is also known as a **parsing error. E.g**

```
print("Welcome to Programming")
    print("Learn Python with us..")
```

- **Incorrect indentation**
- **Missing colon, comma, or brackets**
- **Putting keywords in the wrong place.**

•Logical errors : Even if a statement or expression is syntactically correct, the error that occurs at the **runtime** is known as a **Logical error or Exception**.

- **Indenting a block to the wrong level**
- **Using the wrong variable name**
- **Making a mistake in a boolean expression**

```
a = 10
b = 20
print("Addition:", a + c)
```

RA20 APTT

Built-in Exceptions

Exception	Description
AssertionError	Raised when an assert statement fails.
AttributeError	Raised when attribute assignment or reference fails.
EOFError	Raised when the input() function hits the end-of-file condition.
FloatingPointError	Raised when a floating-point operation fails.
GeneratorExit	Raise when a generator's close() method is called.
ImportError	Raised when the imported module is not found.
IndexError	Raised when the index of a sequence is out of range.
KeyError	Raised when a key is not found in a dictionary.
KeyboardInterrupt	Raised when the user hits the interrupt key (Ctrl+C or Delete)
MemoryError	Raised when an operation runs out of memory.
NameError	Raised when a variable is not found in the local or global scope.
OSError	Raised when system operation causes system related error.
ReferenceError	Raised when a weak reference proxy is used to access a garbage collected referent.

try and except Block to Handling Exceptions

```
fp = open("test.txt", "r")
if fp:
    print("file is opened successfully")
```

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

```
try :
    # statements in try block
except :
    # executed when exception occured in try block
```

The try block is for risky code that can raise an exception and the except block to handle error raised in a try block.

RA20 APTT

try and except Block to Handling Exceptions

```
a = 10
b = 0
c = a / b
print("a/b = %d" % c)
```

```
Traceback (most recent call last):
  File "E:/demos/exception.py", line 3, in <module>
    c = a / b
ZeroDivisionError: division by zero
```

```
try:
    a = 10
    b = 0
    c = a/b
    print("The answer of a divide by b:", c)
except:
    print("Can't divide with zero. Provide different number")
```

```
Can't divide with zero. Provide different number
```

RA20 APTT

Catching Specific Exceptions

```
try:
    a = int(input("Enter value of a:"))
    b = int(input("Enter value of b:"))
    c = a/b
    print("The answer of a divide by b:", c)
except ValueError:
    print("Entered value is wrong")
except ZeroDivisionError:
    print("Can't divide by zero")
```

Output 1:

```
Enter value of a:Ten
Entered value is wrong
```

Output 2:

```
Enter value of a:10
Enter value of b:0
Can't divide by zero
```

Output 3:

```
Enter value of a:10
Enter value of b:2
The answer of a divide by b: 5.0
```

RA20 APTT

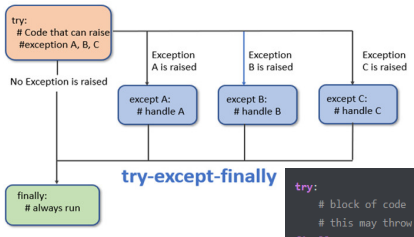
Multiple Exceptions with a single except clause

```
try:
    a = int(input("Enter value of a:"))
    b = int(input("Enter value of b:"))
    c = a / b
    print("The answer of a divide by b:", c)
except(ValueError, ZeroDivisionError):
    print("Please enter a valid value")
```

RA20 APTT

try with finally clause

•The finally block is used to write a block of code that must **execute, whether the try block raises an error or not.**



try-except-finally

```
try:
    # block of code
    # this may throw an exception
finally:
    # block of code
    # this will always be executed
    # after the try and any except block
```

try with finally clause

```
try:
    a = int(input("Enter value of a:"))
    b = int(input("Enter value of b:"))
    c = a / b
    print("The answer of a divide by b:", c)

except ZeroDivisionError:
    print("Can't divide with zero")
finally:
    print("Inside a finally block")
```

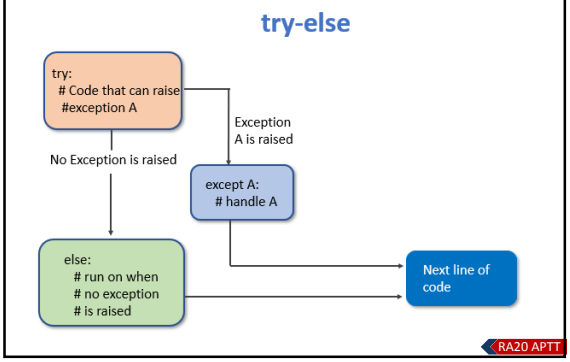
Output 1:

```
Enter value of a:20
Enter value of b:5
The answer of a divide by b: 4.0
Inside a finally block
```

Output 2:

```
Enter value of a:20
Enter value of b:0
Can't divide with zero
Inside a finally block
```

try with Else clause



try with Else clause

```
try:
    a = int(input("Enter value of a:"))
    b = int(input("Enter value of b:"))
    c = a / b
    print("a/b = %d" % c)

except ZeroDivisionError:
    print("Can't divide by zero")
else:
    print("We are in else block ")
```

Output 1

```
Enter value of a: 20
Enter value of b:4
a/b = 5
We are in else block
```

Output 2

```
Enter value of a: 20
Enter value of b:0
Can't divide by zero
```

Raising Exceptions

the raise statement allows us to throw an exception.

```
raise Exception_class,<value>
```

```
def simple_interest(amount, year, rate):
    try:
        if rate > 100:
            raise ValueError(rate)
        interest = (amount * year * rate) / 100
        print('The Simple Interest is', interest)
        return interest
    except ValueError:
        print('interest rate is out of range', rate)

print('Case 1')
simple_interest(800, 6, 8)

print('Case 2')
simple_interest(800, 6, 800)
```

Output:

```
Case 1
The Simple Interest is 384.0

Case 2
interest rate is out of range 800
```

Regular Expression

- A **Regular Expression (RegEx)** is a sequence of characters that defines a search pattern.
- **Regular expression** is used to match, search, replace, and manipulate textual data.
- Some of the cases where regular expressions can help you to save a lot of time.
  - I. **Searching and replacing text in files**
  - II. Validating text input, such as password and email address
  - III. **Rename a hundred files at a time.**
- The **re** module a built-in Python module provides all the required functionality needed for **handling patterns and regular expressions. E.g**

```
^a...s$
```

searching a not as first character and s as not fifth in a string

Regular Expression

```
import re

pattern = '^a...s$'
test_string = 'abyss'
result = re.match(pattern, test_string)

if result:
  print("Search successful.")
else:
  print("Search unsuccessful.")
```

You can change test_string to : abs, **alias,** abyss, **Alias** An abacus and check the output

Meta Characters

•Meta characters are characters that are interpreted in a special way by a RegEx engine. Here's a list of meta characters: **[ ] . ^ $ * + ? {} () \ |.**

Square Bracket [abc] - Square brackets specifies a set of characters you wish to match.

•Here, **[abc]** will match if the string contains any of the **a, b or c** alphabet

we can also specify a range of characters using - inside square brackets.

•[a-e] is the same as [abcde].
•[1-4] is the same as [1234].
•[0-39] is the same as [01239].

RA20 APTT

Meta Characters

•You can complement (invert) the character set by using caret ^ symbol at the start of a square-bracket.
• [^abc] means any character **except a or b or c.**
•[^0-9] means **any non-digit character.**

•. - Period . A period matches any single character (**except newline '\n').**

Expression	String	Matched?
..	a	No match
	ac	1 match
	acd	1 match
	acde	2 matches (contains 4 characters)

RA20 APTT

Meta Characters

^ - Caret: caret symbol ^ is used to check if a string **starts with** a certain character.

Expression	String	Matched?
^a	a	1 match
	abc	1 match
	bac	No match
^ab	abc	1 match
	acb	No match (starts with a but not followed by b)

RA20 APTT

Meta Characters

$ - **Dollar:** $ is used to check if a string **ends with** a certain character.

Expression	String	Matched?
a$	a	1 match
	formula	1 match
	cab	No match

*** - Star:** The star symbol * matches zero or more occurrences of the pattern left to it.

Expression	String	Matched?
ma*n	mn	1 match
	man	1 match
	maaan	1 match
	main	No match (a is not followed by n)
	woman	1 match

APTT

Meta Characters

+ - Plus symbol + matches **one or more occurrences** of the pattern left to it.

Expression	String	Matched?
ma+n	mn	No match (no a character)
	man	1 match
	maaan	1 match
	main	No match (a is not followed by n)
	woman	1 match

? - Question Mark : ? matches **zero or one occurrence** of the pattern left to it.

Expression	String	Matched?
ma?n	mn	1 match
	man	1 match
	maaan	No match (more than one a character)
	main	No match (a is not followed by n)
	woman	1 match

APTT

Meta Characters

{} - Braces: {n,m}. means at least n, and at most m repetitions of the pattern left to it.

Expression	String	Matched?
a{2,3}	abc dat	No match
	abc daat	1 match (at daat)
	aabc daaat	2 matches (at aabc and daaat)
	aabc daaaat	2 matches (at aabc and daaaat)

[0-9]{2, 4} matches at least 2 digits but not more than 4 digits

Expression	String	Matched?
[0-9]{2,4}	ab123csde	1 match (match at ab123csde)
	12 and 345673	3 matches (12, 3456, 73)
	1 and 2	No match

RA20 APTT

Meta Characters

| Alternation: Vertical bar | is used for alternation (or operator).

Expression	String	Matched?
a|b	cde	No match
	ade	1 match (match at ade)
	acdbea	3 matches (at acdbea)

() - Group: Parentheses () is used to group sub-patterns.

Expression	String	Matched?
(a|b|c)xz	ab xz	No match
	abxz	1 match (match at abxz)
	axz cabxz	2 matches (at axzbc cabxz)

Meta Characters

\- Backslash: used to escape various characters including all met characters.

\A - Matches if the specified characters are at the start of a string.

Expression	String	Matched?
\Athe	the sun	Match
	In the sun	No match

\B - Opposite of \b. Matches if the specified characters are **not** at the beginning or end of a word.

Expression	String	Matched?
\Bfoo	football	No match
	a football	No match
	afootball	Match
foo\B	the foo	No match
	the afoo test	No match
	the afootest	Match

Meta Characters

\d - Matches any decimal digit. Equivalent to [0-9]

Expression	String	Matched?
\d	12abc3	3 matches (at 12abc3)
	Python	No match

\D - Matches any non-decimal digit. Equivalent to [^0-9]

Expression	String	Matched?
\D	1ab34"50	3 matches (at 1ab34"50)
	1345	No match

Meta Characters

\w - Matches any alphanumeric character (digits and alphabets). Equivalent to [a-zA-Z0-9_]. By the way, underscore _ is also considered an alphanumeric character.

Expression	String	Matched?
\w	12&": ;c	3 matches (at 12&": ;c)
	%"> !	No match

\W - Matches any non-alphanumeric character. Equivalent to [^a-zA-Z0-9_]

Expression	String	Matched?
\W	1a2%c	1 match (at 1a2%c)
	Python	No match

RE Funtions

re.findall() : The **re.findall()** method returns a list of strings containing all matches.

```python
# Program to extract numbers from a string

import re

string = 'hello 12 hi 89. Howdy 34'
pattern = '\d+'

result = re.findall(pattern, string)
print(result)

# Output: ['12', '89', '34']
```

```python
import re

#"^" matches the start of a string
#"w+"matches the alphanumeric character in the string
substr = "In24/7 SR University ,education is fun"
r1 = re.findall(r"^\w+",substr)
print(r1)
```

RE Funtions

re.search(): takes two arguments: a pattern and a string. The method looks for the first location where the **RegEx pattern** produces a match with the string.

```python
import re
txt = "Yesterday it was raining in warnagal"
x = re.search("\s", txt)
print("The first white-space character is located in position:", x.start())
txt = "The rain in Spain"
x = re.search("Telangana", txt)
print(x)
```

```python
import re

string = "Python is fun"

# check if 'Python' is at the beginning
match = re.search('\APython', string)

if match:
  print("pattern found inside the string")
else:
  print("pattern not found")

# Output: pattern found inside the string
```

RE Funtions

re.split(): The **re.split method splits** the string where there is a match and returns a list of strings where the splits have occurred.

```
import re

string = 'Twelve:12 Eighty nine:89.'
pattern = '\d+'

result = re.split(pattern, string)
print(result)

# Output: ['Twelve:', ' Eighty nine:', '.']
```

```
import re

string = 'Twelve:12 Eighty nine:89 Nine:9.'
pattern = '\d+'

# maxsplit = 1
# split only at the first occurrence
result = re.split(pattern, string, 1)
print(result)

# Output: ['Twelve:', ' Eighty nine:89 Nine:9.']
```

RE Funtions

```
import re
txt = "The rain in Telangana"
x = re.split("\s", txt) #Split at each white-space character
print(x)
txt = "The rain in Telangana"
x = re.split("\s", txt, 1) #Split the string only at the first occurrence
print(x)
```

```
import re
txt = "The rain in Warangal"
x = re.sub("\s", "==", txt)
print(x)
txt = "The rain in Waranagl"
x = re.sub("\s", "**", txt, 1)
print(x)
```

Enumerate

•**Enumerate** The enumerate() function is useful when **we wanted to access both value and its index number or any sequence such as list or string.**

•The enumerate() method adds a counter to an iterable and returns it (the enumerate object).

•enumerate() method takes two parameters:

I. **iterable** - a sequence, an iterator, or objects that supports iteration

II. **start** (optional) - enumerate() starts counting from this number if start is omitted, 0 is taken as start.

RA20 APTT

Enumerate

```
languages = ['Python', 'Java', 'JavaScript']

# convert enumerate object to list
print(list(enumerate(languages)))
```

```
numbers = [4, 2, 5, 7, 8]
for i, v in enumerate(numbers):
    print('At Index[', i, '] NUmber =', v)
```

Enumerate

```
grocery = ['bread', 'milk', 'butter', 'Sugar', 'Salt']
EG = enumerate(grocery)
print(type(EG))

# converting to list
print(list(EG))

# changing the default counter
EG = enumerate(grocery, 101)
print(list(EG))
```

RA20 APTT

Looping over Enumerate

```
grocery = ['bread', 'milk', 'butter', 'Sugar', 'Salt']
for item in enumerate(grocery):
    print(item)

for count, item in enumerate(grocery):
  print(count, item)

for count, item in enumerate(grocery, 101):
  print(count, item)
```

RA20 APTT

Anonymous or Lambda Function

•In Python, an anonymous function is a function that is defined without a name.

•While normal functions are defined using the def keyword in Python, anonymous functions are defined using the lambda keyword.

•Lambda functions can have any number of arguments but only one expression.

•The expression is evaluated and returned.

• Lambda functions can be used wherever function objects are required.

•Lambda functions are used along with built-in functions like filter(), map() etc.

```
lambda arguments: expression
```

RA20 APTT

Anonymous or Lambda Function

```
double = lambda x : x * 2

print(double(5))
```

```
def double(x):
    return x * 2

print(double(5))
```

•Here, **lambda x: x * 2** is the **lambda function**.

•Here x is the argument and x * 2 is the expression that gets evaluated and returned.

This function has no name and returns a function object which is assigned to the **identifier double**.

We use lambda functions when we require a nameless function for a short period of time

RA20 APTT

Anonymous or Lambda Function

```
x = lambda a : a + 10
print(x(5))

x = lambda a, b : a * b
print(x(5, 6))

x = lambda a, b, c : a + b + c
print(x(5, 6, 2))
```

```
def myfunc(n):
  return lambda a : a * n

doubl = myfunc(2)

print(doubl(11))
```

use Lambda as an anonymous function inside another function.

RA20 APTT

Anonymous or Lambda Function

•The filter() function in Python takes in a function and a list as arguments.

```
#Program to filter out only the even items from a list
my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(filter(lambda x: (x%2 == 0), my_list))

print(new_list)
```

```
age_group = [5, 12, 17, 18, 24, 32, 43, 14]

def myFunc(x):
  if x < 18:
    return False
  else:
    return True

adults = filter(myFunc, age_group)

for x in adults:
  print(x)
```

Anonymous or Lambda Function

•The map() function returns a map object of the results after applying the given function to each item of a given iterable (list, tuple etc.)

```
#Program to double each item in a list using map()

my_list = [1, 5, 4, 6, 8, 11, 3, 12]
new_list = list(map(lambda x: x * 2, my_list))
print(new_list)
```

```
def calculateSquare(n):
    return n*n

numbers = (1, 2, 3, 4)
result = map(calculateSquare, numbers)
print(result)

# converting map object to set
numbersSquare = set(result)
print(numbersSquare)
```

Anonymous or Lambda Function

```
num1 = [4, 5, 6]
num2 = [5, 6, 7]

result = map(lambda n1, n2: n1+n2, num1, num2)
print(list(result))
```

RA20 APTT

Anonymous or Lambda Function

```
numbers = [2, 4, 6, 8, 10]

# returns square of a number
def square(number):
  return number * number

# apply square() function to each item of the numbers list
squared_numbers_iterator = map(square, numbers)

# converting to list
squared_numbers = list(squared_numbers_iterator)
print(squared_numbers)
```

RA20 APTT

Anonymous or Lambda Function

•The reduce() function: It performs a rolling-computation as specified by the passed function to the neighboring elements, by taking a *function* and an *iterable* as arguments, and returns the *final* computed value.

```
from functools import *
# Returns the sum of all the elements using `reduce`
result = reduce((lambda a, b: a + b), [1, 2, 3, 4])
print(result)
```

Using Lambda.

```
from functools import *

def summation(a,b):
    return a+b

result = reduce(summation, [1, 2, 3, 4])
print(result)
```

Using Pre-Defined Function

RA20 APTT