1. **Implement a Python class MaxHeap that supports the following operations: insert, delete, and get\_max. Ensure the operations maintain the properties of a max-heap.**

class MaxHeap:

List = []

def insert(number):

MaxHeap.List.append(number)

def delete(number):

MaxHeap.List.remove(number)

def get\_max():

Max = 0

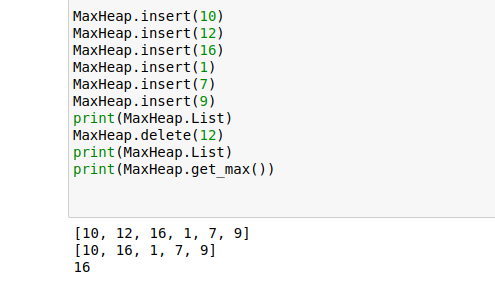
for i in MaxHeap.List:

if i>Max:

Max = i

return Max

**Output:**



1. **Write a Python function that takes a list of URLs, attempts to download their content, and retries up to 3 times if an error occurs. Use appropriate error handling to manage different types of exceptions.**

import webbrowser

def download(urls):

for k in urls:

try:

for i in range(3):

if webbrowser.open(k):

break

except:

"Could not download!"

1. **Write a Python script that trains a simple linear regression model using scikit-learn. Use a dataset of your choice, split it into training and testing sets, and evaluate the model's performance.**
2. **Using pandas, write a Python function to clean and preprocess a given DataFrame, which involves handling missing values, normalizing numerical columns, and encoding categorical columns.**
3. **Write a Python function to compute the nth Fibonacci number using recursion.**

def get\_fibonacci(n):

Sum = 1

series = [1]

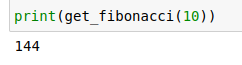
for i in range(n):

series.append(Sum)

Sum+=series[i]

return Sum

**Output:**



1. **Write a Python function that divides two numbers and handles the case where the divisor is zero by returning a custom error message.**

def division(dividend, divisor):

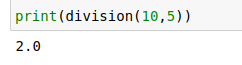
if divisor == 0:

return "Cannot divide by zero!"

else:

return dividend/divisor

**Output:**



1. **Write a Python decorator that measures the execution time of a function and logs it. Apply this decorator to a function that performs a computationally expensive task.**

1. **Write a Python function that takes two numbers and an operator (as a string) and performs the corresponding arithmetic operation (addition, subtraction, multiplication, or division).**

def calculator(num1,operation,num2):

if operation == "+":

return num1+num2

elif operation == "-":

return num1-num2

elif operation == "\*":

return num1\*num2

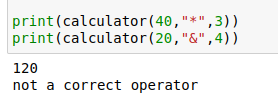
elif operation == "/":

return num1/num2

else:

return "not a correct operator"

**Output:**



1. **Write a Python function that generates a random password. The password should contain a mix of uppercase letters, lowercase letters, digits, and special characters.**

import random as rd

def random\_password\_generator():

length = rd.randint(8,20)

password = []

Final = ""

Choice = ["Uppercase","Lowercase","numbers","special characters"]

for i in range(length):

choice = rd.choice(Choice)

if choice == "Uppercase":

password.append(chr(rd.randint(65,90)))

elif choice == "Lowercase":

password.append(chr(rd.randint(97,122)))

elif choice == "numbers":

password.append(chr(rd.randint(48,57)))

else:

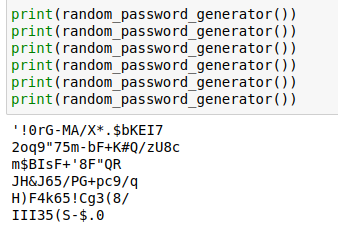
password.append(chr(rd.randint(33,47)) or chr(rd.randint(58,64)) or chr(rd.randint(91,96)) or chr(rd.randint(123,126)))

for j in password:

Final+=j

return Final

**Output:**



1. **Write a Python function that takes a 2D list (matrix) and returns its transpose.**

import numpy as np

def transpose(matrix):

new\_matrix = []

for i in range(matrix.shape[1]):

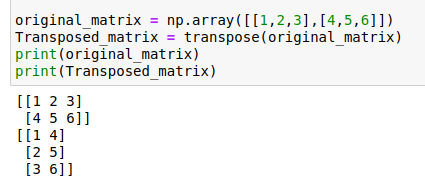
for j in range(matrix.shape[0]):

new\_matrix.append(matrix[j][i])

Final = np.array(new\_matrix)

Final = Final.reshape(matrix.shape[1],matrix.shape[0])

return Final

**Output:**