|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
|  | Practical FileComputer Science(Class XII) |  |

|  |  |
| --- | --- |
| Submitted To  Mr. A.K. Mishra  Headmaster Sr. Sch. | Submitted By  Gayatri P  XII Sci |

# Python

# Assignment 1

***Objective****:* Write a function that accepts a length written in feet as an argument and returns this length written in inches. Write a second function that asks the user for a number of feet and returns this value. Write a third function that accepts a number of inches and displays this to the screen. Use these three functions to write a program that asks the user for a number of feet and tells them the corresponding number of inches. (1 foot = 12 inches)

***Code***:

def number\_of\_feet(feet):

    return feet

def feet\_into\_inches(feet):

    inches=feet\*12

    return inches

def number\_of\_inches(inches):

    return inches

feet=float(input("Enter height in feet : "))

Inches= number\_of\_inches(feet\_into\_inches(number\_of\_feet(feet)))

print("No. of inches =",Inches ,"inches")

***Input****:*

Enter height in feet : 12

***Output****:*

No. of inches = 144.0 inches

# Assignment 2

***Objective***: Input any number from user and calculate the factorial of the number.

***Code****:*

num = int(input("Enter any number : "))

fact = 1

n = num # storing num in n for printing

while num > 1: # loop to iterate from n to 2

    fact \*= num

    num -= 1

print(f'Factorial of {n} is : {fact}')

***Input****:*

Enter any number : 8

***Output****:*

Factorial of 8 is : 40320

***Input****:*

Enter any number : 5

***Output****:*

Factorial of 8 is : 120

# Assignment 3

***Objective***: Input any number from the user and check if it’s a prime number or not.

***Code****:*

num = int(input("Enter any number : "))

def isPrime(n):

    if n < 2:

        return False

    for i in range(2,int(num\*\*0.5)+1):

        if num % i == 0:

            return False

    return True

if isPrime(num):

    print(f'{num} is a prime number.')

else:

    print(f'{num} is not a prime number.')

***Input****:*

Enter any number : 71

***Output****:*

71 is a prime number.

***Input****:*

Enter any number : 69

***Output****:*

69 is not a prime number.

# Assignment 4

***Objective***: Write a program to check if a given string is a palindrome or not.

***Code****:*

s = input('Enter a string: ')

def isPalindrome(s):

    if s == s[::-1]:

        return True

    return False

if isPalindrome(s):

    print(s, 'is a Palindrome')

else:

    print(s, 'is not a Palindrome')

***Input****:*

Enter a string: racecar

***Output****:*

racecar is a Palindrome

***Input****:*

Enter a string: yellow

***Output****:*

yellow is not a Palindrome

# Assignment 5

***Objective***: Write a guessing game where the user has to guess the correct number. The computer should give hints whether the guessed number is greater or smaller than the correct number. It should also records the number of incorrect tries.

***Code****:*

import random

start, end, tries = 0, 100, 1

n = random.randint(start, end)

print('-----Guessing Game-----')

print(f'The computer has guessed an integer between {start} and {end}.')

guess = int(input('Guess the number: '))

while guess != n:

    tries += 1

    if n > guess:

        print('The number is larger than you think.')

    elif n < guess:

        print('The number is smaller than you think.')

    guess = int(input('Guess again: '))

print(f'Yes the  number is {n}. You have guessed it correctly in {tries} tries.')

***Output****:*

-----Guessing Game-----

The computer has guessed an integer between 0 and 100.

Guess the number: 50

The number is smaller than you think.

Guess again: 20

The number is smaller than you think.

Guess again: 16

Yes the number is 16. You have guessed it correctly in 3 tries.

# Assignment 6

***Objective***: Write a program which converts a given decimal number into a roman numeral.

***Code****:*

romanKeys = {

    'M': 1000,

    'CM': 900,

    'D': 500,

    'CD': 400,

    'C': 100,

    'XC': 90,

    'L': 50,

    'XL': 40,

    'X': 10,

    'IX': 9,

    'V': 5,

    'IV': 4,

    'I': 1

}

def toRoman(n):

    roman = ''

    for key in romanKeys:

        while n >= romanKeys[key]:

            roman += key

            n -= romanKeys[key]

    return roman

n = int(input('Enter a number: '))

print(toRoman(n))

***Input****:*

Enter a number: 45

***Output****:*

XLV

***Input****:*

Enter a number: 789

***Output****:*

DCCLXXXIX

# Assignment 7

***Objective***: Create a module 'mensuration.py' that stores function to find area and perimeter of following figures:

(a)Circle, (b)Square, (c)Rectangle, (d)Equilateral Triangle, (e)Parallelogram.

Insert help() function, which should give proper information about module.

***Code****:*

"""mensuation.py - Mensuration Functions for calculating perimeter and area of

(a)Circle, (b)Square, (c)Rectangle, (d)Equilateral Triangle, (e)Parallelogram .

NOTE: IN ALL THE FUNCTIONS, YOU HAVE TO SPECIFY WHETHER YOU WANT TO CALCULATE

   (i)  AREA OR,

   (ii) PERIMETER

DURING FUNCTION-CALL IN THE SAME FORMAT AS SPECIFIED IN THE FUNCTION'S PARAMETER

"""

def rectangle(length,breadth,choice\_for\_perimeter\_or\_area):

    """Print either area or perimeter of rectangle,

    on the basis of input given to "choice\_for\_perimeter\_or\_area"."""

    if choice\_for\_perimeter\_or\_area=="area":

        area=length\*breadth

        print(area)

    elif choice\_for\_perimeter\_or\_area=="perimeter":

        perimeter=2\*(length+breadth)

        print(perimeter)

def square(side,choice\_for\_perimeter\_or\_area):

    """Print either area or perimeter of square,

   on the basis of input given to "choice\_for\_perimeter\_or\_area"."""

    if choice\_for\_perimeter\_or\_area=="area":

        area=side\*\*2

        print(area)

    elif choice\_for\_perimeter\_or\_area=="perimeter":

        perimeter=4\*side

        print(perimeter)

def circle(radius,choice\_for\_perimeter\_or\_area):

    """Print either area or perimeter of circle,

    on the basis of input given to "choice\_for\_perimeter\_or\_area"."""

    if choice\_for\_perimeter\_or\_area=="area":

        area=(22/7)\*(radius\*\*2)

        print(area)

    elif choice\_for\_perimeter\_or\_area=="perimeter":

        perimeter=2\*(22/7)\*radius

        print(perimeter)

def equilateral\_triangle(side,choice\_for\_perimeter\_or\_area):

    """Print either area or perimeter of equilatereal triangle,

    on the basis of input given to "choice\_for\_perimeter\_or\_area"."""

    if choice\_for\_perimeter\_or\_area=="area":

        area=((3\*\*(0.5))/4)\*(side\*\*2)

        print(area)

    elif choice\_for\_perimeter\_or\_area=="perimeter":

        perimeter=3\*side

        print(perimeter)

def parallelogram(base,height,other\_side,choice\_for\_perimeter\_or\_area):

    """Print either area or perimeter of parallelogram,

    on the basis of input given to "choice\_for\_perimeter\_or\_area"."""

    if choice\_for\_perimeter\_or\_area=="area":

        area=base\*height

        print(area)

    elif choice\_for\_perimeter\_or\_area=="perimeter":

        perimeter=2\*(base+other\_side)

        print(perimeter)

***Output****:*

>>> import mensuration

>>> help(mensuration)

Help on module mensuration:

NAME

mensuration

DESCRIPTION

mensuation.py - Mensuration Functions for calculating perimeter and area of

(a)Circle, (b)Square, (c)Rectangle, (d)Equilateral Triangle, (e)Parallelogram .

NOTE: IN ALL THE FUNCTIONS, YOU HAVE TO SPECIFY WHETHER YOU WANT TO CALCULATE

(i) AREA OR,

(ii) PERIMETER

DURING FUNCTION-CALL IN THE SAME FORMAT AS SPECIFIED IN THE FUNCTION'S PARAMETER

FUNCTIONS

circle(radius, choice\_for\_perimeter\_or\_area)

Print either area or perimeter of circle,

on the basis of input given to "choice\_for\_perimeter\_or\_area".

equilateral\_triangle(side, choice\_for\_perimeter\_or\_area)

Print either area or perimeter of equilatereal triangle,

on the basis of input given to "choice\_for\_perimeter\_or\_area".

parallelogram(base, height, other\_side, choice\_for\_perimeter\_or\_area)

Print either area or perimeter of parallelogram,

on the basis of input given to "choice\_for\_perimeter\_or\_area".

rectangle(length, breadth, choice\_for\_perimeter\_or\_area)

Print either area or perimeter of rectangle,

on the basis of input given to "choice\_for\_perimeter\_or\_area".

square(side, choice\_for\_perimeter\_or\_area)

Print either area or perimeter of square,

on the basis of input given to "choice\_for\_perimeter\_or\_area".

FILE

f:\python\test\mensuration.py

>>> mensuration.circle(6, 'perimeter')

37.714285714285715

>>> mensuration.equilateral\_triangle(10, 'area')

43.30127018922193

# Assignment 8

***Objective***: Create a module ‘calculator.py’ which can perform addition, subtraction, multiplication and division on given numbers.

***Code****:*

'''

Performs simple calulation operations

'''

def add(\*numbers):

    '''Returns the sum of a list of numbers passed as the argument.

    Eg: add(1, 2, 3, 4) will return 10'''

    return sum(numbers)

def subtract(number\_1, number\_2):

    '''Returns the difference of number\_1 and number\_2.

    Eg: add(6, 2) will return 4'''

    return number\_1 - number\_2

def multiply(\*numbers):

    '''Returns the product of a list of numbers passed as the argument.

    Eg: add(1, 2, 3, 4) will return 24'''

    p = 1

    for n in numbers:

        p \*= n

    return p

def divide(divident, divisor):

    '''Returns the quotient of divident and divident.

    Eg: add(6, 2) will return 3.0'''

    return divident / divisor

***Output in IDLE****:*

>>> import calculator

>>> help(calculator)

Help on module calculator:

NAME

calculator - Performs simple calulation operations

FUNCTIONS

add(\*numbers)

Returns the sum of a list of numbers passed as the argument.

Eg: add(1, 2, 3, 4) will return 10

divide(divident, divisor)

Returns the quotient of dividend and divisor.

Eg: add(6, 2) will return 3.0

multiply(\*numbers)

Returns the product of a list of numbers passed as the argument.

Eg: add(1, 2, 3, 4) will return 24

subtract(number\_1, number\_2)

Returns the difference of number\_1 and number\_2.

Eg: add(6, 2) will return 4

FILE

f:\python\test\calculator.py

>>> calculator.multiply(2, 556, 432)

480384

>>> calculator.divide(51, 17)

3.0

>>> calculator.add(1, 2, 3)

6

>>> calculator.subtract(51, 6)

45

# Assignment 9

***Objective***: Write a program to implement a stack in python using lists.

***Code****:*

def push(item):

    stack.append(item)

def pop():

    if stack:

        print(stack.pop(), ' deleted')

    else:

        print('Underflow')

def peek():

    if stack:

        print('Top:', stack[-1])

    else:

        print('Underflow')

def display():

    if stack:

        print('\n'.join(map(str, stack)), end=' ')

        print('(Top)')

    else:

        print('Underflow')

print("\*\*\*\* STACK DEMONSTRATION \*\*\*\*\*\*")

print("1. PUSH ")

print("2. POP")

print("3. PEEK")

print("4. SHOW STACK ")

print("0. EXIT\n")

stack = []

while True:

    ch = int(input("Enter your choice : "))

    if ch == 1:

        push(int(input("Enter Item to Push : ")))

    elif ch==2:

        pop()

    elif ch==3:

        peek()

    elif ch==4:

        display()

    elif ch == 0:

        break

    else:

        print('Invalid Choice')

    print()

***Output (Input in blue)****:*

\*\*\*\* STACK DEMONSTRATION \*\*\*\*\*\*

1. PUSH

2. POP

3. PEEK

4. SHOW STACK

0. EXIT

Enter your choice : 1

Enter Item to Push : 2

Enter your choice : 1

Enter Item to Push : 4

Enter your choice : 1

Enter Item to Push : 6

Enter your choice : 3

Top: 6

Enter your choice : 4

2

4

6 (Top)

Enter your choice : 2

6 deleted

Enter your choice : 4

2

4 (Top)

Enter your choice : 0

# Assignment 10

***Objective***: Write a program that implements a stack where the elements are shifted towards right so that the top always remains at 0th index.

***Code****:*

def Push(stack, item):

    stack.insert(0, item)

    return stack

def Pop(stack):

    a = stack[0]

    stack.remove(a)

    return a

def Peek(stack):

    print('Top: ', stack[0])

print("\*\*\*\* STACK DEMONSTRATION \*\*\*\*\*\*")

print("1. PUSH ")

print("2. POP")

print("3. PEEK")

print("0. EXIT")

stack = []

while True:

    ch = int(input("Enter your choice : "))

    if ch==1:

        v = Push(stack, int(input("Enter Item to Push : ")) )

        print('New stack: ', v)

    elif ch==2:

        v = Pop(stack)

        print('Deleted item: ', v)

    elif ch==3:

        val = Peek(stack)

    print()

***Output****:*

\*\*\*\* STACK DEMONSTRATION \*\*\*\*\*\*

1. PUSH

2. POP

3. PEEK

Enter your choice : 1

Enter Item to Push : 4

New stack: [4]

Enter your choice : 1

Enter Item to Push : 6

New stack: [6, 4]

Enter your choice : 1

Enter Item to Push : 8

New stack: [8, 6, 4]

Enter your choice : 2

Deleted item: 8

Enter your choice : 3

Top: 6

Enter your choice : 0

# Assignment 11

***Objective***: Write the game 2048 using stacks.

Input: A list of integers.

Output: If two adjacent numbers are equal, the will be merged into one number with double the value. The task is to find the final set of numbers so that they cannot be merged further.

Eg: The input [2, 2, 4, 8, 8] will give [8, 16]

***Code****:*

l = eval(input('Enter list: '))

stack = [l[0]]

for i in range(1, len(l)):

    stack.append(l[i])

    stack.sort()

    if len(stack) >= 2:

        if stack[-1] == stack[-2]:

            stack.pop()

            a = stack.pop()

            stack.append(a\*2)

print('Final Set: ', stack)

***Input****:*

Enter list: [2, 2, 4, 8, 8]

***Output****:*

Final Set: [8, 16]

***Input****:*

Enter list: [2, 2, 4, 8, 16]

***Output****:*

Final Set: [32]

***Input****:*

Enter list: [2, 4, 4, 8, 8, 16, 32, 32]

***Output****:*

Final Set: [2, 8, 32, 64]

# Assignment 12

***Objective***: Write a function shiftArray(array, n) which takes an array and shifts its elementsleft by n places

***Code****:*

a = eval(input('Enter an array:' ))

n = int(input('Enter n: '))

def shiftArray(array, n):

    a, b = array[n:], array[:n]

    return a+b

print(shiftArray(a, n))

***Input****:*

Enter an array:[1, 2, 3, 4, 5, 6]

Enter n: 1

***Output****:*

[2, 3, 4, 5, 6, 1]

***Input****:*

Enter an array:[1, 2, 3, 4, 5, 6]

Enter n: 4

***Output****:*

[5, 6, 1, 2, 3, 4]

# Assignment 13

***Objective***: Write a program that takes two matrices as input and adds them. Then print the result

***Code****:*

M1 = eval(input('Enter a matrix: '))

M2 = eval(input('Enter a matrix: '))

def add(M1, M2):

    S = [[] for i in range(len(M1))]

    for i, (row1, row2) in enumerate(zip(M1, M2)):

        for a, b in zip(row1, row2):

            S[i].append(a+b)

    return S

S = add(M1, M2)

for row in S:

    print(row)

***Input****:*

Enter a matrix: [[4, 5, 18], [2, 5, 13], [1, 18, 20]]

Enter a matrix: [[8, 8, 18], [7, 12, 14], [19, 22, 4]]

***Output****:*

[12, 13, 36]

[9, 17, 27]

[20, 40, 24]

# Assignment 14

***Objective***: Write a program that calculated the HCF (Highest Common Factor) and LCM (Lowest Common Denominator) of two numbers.

***Code****:*

x = int(input('Enter 1st number: '))

y = int(input('Enter 2nd number: '))

a = max(x, y)

b = min(x, y)

while a % b != 0:

    a, b = b, (a % b)

hcf = b

lcm = int((x\*y) / hcf)

print(f'HCF of {x} and {y} : {hcf}')

print(f'LCM of {x} and {y} : {lcm}')

***Input****:*

Enter 1st number: 51

Enter 2nd number: 17

***Output****:*

HCF of 51 and 17 : 17

LCM of 51 and 17 : 51

***Input****:*

Enter 1st number: 16

Enter 2nd number: 90

***Output****:*

HCF of 51 and 17 : 2

LCM of 51 and 17 : 720

# Assignment 15

***Objective***: Write a program that checks if two words are anagrams. An anagram is a word/phrase formed by rearranging the letters of another word.

***Code****:*

s1 = input('Enter word 1: ')

s2 = input('Enter word 2: ')

def isAnagram(s1, s2):

    letters1, letters2 = {}, {}

    if len(s1) == len(s1):

        # dictionary to count the number of letters

        for x in s1:

            letters1[x] = letters1[x]+1 if x in letters1 else 1

        for x in s2:

            letters2[x] = letters2[x]+1 if x in letters2 else 1

        return letters1 == letters2

    return False

if isAnagram(s1, s2):

    print(f'{s1} & {s2} are anagrams.')

else:

    print(f'{s1} & {s2} are not anagrams.')

***Input****:*

Enter word 1: listen

Enter word 2: silent

***Output****:*

listen & silent are anagrams.

***Input****:*

Enter word 1: rose

Enter word 2: sure

***Output****:*

rose & sure are not anagrams.

# Assignment 16

***Objective***: Write a program that generates an array made up of random numbers (between 1 to 100) and sorts the array using bubble sort.

***Code****:*

from random import randint

l = [randint(1, 100) for i in range(10)]

print('Original: ', l)

n = len(l)

for i in range(n):

    for j in range(n-i-1):

        if l[j+1] < l[j]:

            l[j+1], l[j] = l[j], l[j+1]

print('Sorted: ', l)

***Output 1****:*

Original: [45, 50, 16, 100, 16, 56, 78, 47, 31, 96]

Sorted: [16, 16, 31, 45, 47, 50, 56, 78, 96, 100]

***Output 2****:*

Original: [22, 84, 2, 22, 11, 28, 68, 88, 37, 80]

Sorted: [2, 11, 22, 22, 28, 37, 68, 80, 84, 88]

# Assignment 17

***Objective***: Write a program that prompts the user for an angle in degrees and estimates the value of sin(x) using the expansion upto n = 10.

***Code****:*

deg = int(input('Enter degree: '))

a = (3.14159265/180)\*deg

x = 3 # power

def factorial(n): # factorial

    pr = 1

    for i in range(n, 0, -1):

        pr \*= i

    return pr

def sin(n):

    val = a

    x = 3

    for i in range(10):

        if i%2 == 0:

            val -= ((n\*\*x)/factorial(x))

        else:

            val += ((n\*\*x)/factorial(x))

        x += 2

    return val

print(f'sin({deg}) = {sin(a)}')

***Input****:*

Enter degree: 30

***Output****:*

sin(30) = 0.49999999948185797

***Input****:*

Enter degree: 90

***Output****:*

sin(90) = 1.0

# Assignment 18

***Objective***: Write a program to read and display file content line by line with each word separated by '#'

***Code****:*

f = open('t.txt')

lines = f.readlines()

for line in lines:

    a = line.replace(' ', '#')

    print(a.strip())

f.close()

***Original Content of the File****:*

India is my country

I love python

Python learning is fun

***Output****:*

India#is#my#country#

I#love#python#

Python#learning#is#fun#

# Assignment 19

***Objective***: Write a program to read the content of file and display the total number of consonants, uppercase, vowels and lower case characters.

***Code****:*

v = c = u = l = o = 0

f = open('t.txt')

data = f.read()

f.close()

vowels=['a','e','i','o','u']

for ch in data:

    if ch.isalpha():

        if ch.lower() in vowels:

            v += 1

        else:

            c += 1

    if ch.isupper():

        u += 1

    elif ch.islower():

        l += 1

    elif ch != ' ' and ch != '\n':

        o += 1

print('Number of vowels:\t\t', v)

print('Number of consonants:\t\t', c)

print('Number of capital letters:\t', u)

print('Number of small letters:\t', l)

print('Number of other characters:\t', o)

***Contents of the File****:*

India is my country

Python learning is fun

345@32

12345

***Output****:*

Number of vowels: 12

Number of consonants: 23

Number of capital letters: 2

Number of small letters: 33

Number of other characters: 11

# Assignment 20

***Objective***: Write a program to take 10 sample phishing emails, and find the most common word occurring.

***Code****:*

emails = [

    "jackpotwin@lottery.com",

    "claimtheprize@mymoney.com",

    "youarethewinner@lottery.com",

    "luckywinner@mymoney.com",

    "spinthewheel@flipkart.com",

    "dealwinner@snapdeal.com"

    "luckywinner@snapdeal.com"

    "luckyjackpot@americanlottery.com"

    "claimtheprize@lootolottery.com"

    "youarelucky@mymoney.com",

]

d={}

for e in emails:

    x = e.split('@')

    for w in x:

        d[w] = d[w] + 1 if w in d else 1

key\_max = max(d, key=d.get)

print("Most Common Occuring word is :", key\_max)

***Output****:*

Most Common Occuring word is : mymoney.com

# Assignment 21

***Objective***: Write a program to read the content of file line by line and write it to another file except for the lines contains 'a' letter in it.

***Code****:*

f1 = open("file2.txt")

f2 = open("file2copy.txt","w")

for line in f1:

    if 'a' not in line:

        f2.write(line)

print('File Copied Successfully!')

f1.close()

f2.close()

***Contents of the original file****:*

a quick brown fox

one two three four

five six seven

India is my country

eight nine ten

bye!

***Contents of the copied file****:*

one two three four

five six seven

eight nine ten

bye!

# Assignment 22

***Objective***: Write a program to find the frequency of a given word in the file.

***Code****:*

word = input('Enter word to search: ')

c = 0

with open('t.txt') as f:

    line = f.readline()

    while line:

        if word in line:

            c += line.count(word)

        line = f.readline()

print(f'There are {c} occurence(s) of {word} in the file.')

***Contents of the file****:*

A computer network is a collection of interconnected computers and other devices which are able to communicate with each other. Also defined as - collection of hardware components and computers interconnected by communication channels that allow sharing of resources and information. Where at least one process in one device is able to send/receive data to/from at least one process residing in a remote device, then the two devices are said to be in a network

***Input****:*

Enter word to search: the

***Output****:*

There are 4 occurence(s) of the in the file.

***Input****:*

Enter word to search: computers

***Output****:*

There are 2 occurence(s) of the in the file.

# Assignment 23

***Objective***: Write a program to append employee details (eid, name, salary) into an existing csv file.

***Code****:*

import csv

with open('employee.csv', 'a') as file:

    csv\_writer = csv.writer(file)

    ans = 'y'

    while ans.lower() == 'y':

        eid = int(input('Enter employee id: '))

        name = input('Enter employee name: ')

        salary = input('Enter employee salary: ')

        csv\_writer.writerow([eid, name, salary])

        ans = input('Add more? (y/n): ')

***Output****:*

Enter employee id: 1

Enter employee name: jon

Enter employee salary: 8000

Add more? (y/n): y

Enter employee id: 2

Enter employee name: sandra

Enter employee salary: 7000

Add more? (y/n): y

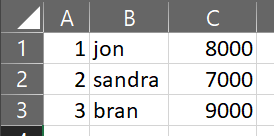
Enter employee id: 3

Enter employee name: bran

Enter employee salary: 9000

Add more? (y/n): n

***File****:*



# Assignment 24

***Objective***: A file contains a list of telephone numbers along with names in the format:

Robert, 9293194935

Arvind, 7891838283…

Write a program to write them in tabular format along with serial number.

***Code****:*

import csv

with open('employee.csv', 'r') as file:

    csv\_reader = csv.reader(file)

    print('S.no.\tName\tPhone no.')

    for i, row in enumerate(csv\_reader):

        print(f'{i+1}\t{row[0]}\t{row[1]}')

***Output****:*

S.no. Name Phone no.

1 Robert 9293194935

2 Arvind 3423423424

3 Raj 2872346283

4 Hemant 1231231423

# Assignment 25

***Objective***: Write a Python program to read a list of employee data (eid, name, dept) and print all the unique departments.

***Code****:*

import csv

depts = []

with open('employee.csv', 'r') as file:

    csv\_reader = csv.reader(file)

    print('All departments:')

    for row in csv\_reader:

        dept = row[2]

        if dept not in depts: #check if already in depts

            print(dept)

            depts.append(dept) #adding to printed depts

***File****:*

1,Robert,IT

2,Arvind,Admin

3,Raj,Sales

4,Hemant,IT

5,Yogesh,Sales

6,Anant,HDR

***Output****:*

All departments:

IT

Admin

Sales

HDR

# Assignment 26

***Objective***: Write a program to write student roll no, name and marks to a binary file.

***Code****:*

import pickle

with open('student.dat','ab') as f:

    ans='y'

    while ans.lower()=='y':

        roll = int(input("Enter Roll Number : "))

        name = input("Enter Name : ")

        marks = int(input("Enter Marks : "))

        pickle.dump([roll,name,marks], f)

        ans=input("Add More ?(Y) ")

        print()

***Output****:*

Enter Roll Number : 1

Enter Name : jon

Enter Marks : 89

Add More ?(Y) y

Enter Roll Number : 3

Enter Name : meera

Enter Marks : 90

Add More ?(Y) n

Enter Roll Number : 9

Enter Name : sam

Enter Marks : 100

Add More ?(Y) n

# Assignment 27

***Objective***: Write a program to update student marks in a binary file using the roll number and show error if the given roll no. is not found.

***Code****:*

import pickle

roll = int(input('Enter roll no to update: '))

records = []

found = False

with open('student.dat','rb') as f:

    while True:

        try:

            record = pickle.load(f)

            records.append(record)

        except:

            break

with open(r'test\student.dat','wb') as f:

    for record in records:

        if record[0] == roll:

            print('Record found\nName: ', record[1])

            print('Marks: ', record[2])

            marks = int(input('Enter new marks: '))

            record[2] = marks

            found = True

            print('\nRecord Updated')

        pickle.dump(record, f)

    if not(found):

        print('Record cannot be found.')

***Output 1****:*

Enter roll no to update: 9

Record found

Name:  sam

Marks:  100

Enter new marks: 98

Record Updated

***Output 2****:*

Enter roll no to update: 12

Record cannot be found.

# Assignment 28

***Objective***: Write a program to read, write and search students records to a binary file according to the user choice.

***Code****:*

import pickle

file = 'data.pickle'

def insert\_data():

    roll = int(input('Insert roll no.: '))

    name = input('Insert name: ')

    marks = int(input('Insert marks: '))

    record = [roll, name, marks]

    with open(file, 'ab') as f:

        pickle.dump(record, f)

        print('Data inserted')

def read\_records():

    with open(file, 'rb') as f:

        while True:

            try:

                record = pickle.load(f)

                print(' '.join(str(x) for x in record))

            except:

                break

def search\_record():

    roll = int(input('Insert roll no.: '))

    found = False

    with open(file, 'rb') as f:

        while True:

            try:

                record = pickle.load(f)

                if record[0] == roll:

                    print(f'\nRecord found:\nName: {record[1]}\nMarks: {record[2]}')

                    found = True

            except:

                if not(found):

                    print('\nNot found')

                break

ans = 'y'

while ans == 'y':

    print('\n\*\*\*STUDENT DATABASE\*\*\*')

    print('1. Insert data')

    print('2. Search data')

    print('3. Read all data')

    s = int(input('Selection: '))

    if s == 1:

        insert\_data()

    elif s == 2:

        search\_record()

    else:

        read\_records()

    ans = input('\nWant to continue? y/n ')

***Output****:*

\*\*\*STUDENT DATABASE\*\*\*

1. Insert data

2. Search data

3. Read all data

Selection: 1

Insert roll no.: 1

Insert name: bran

Insert marks: 72

Data inserted

Want to continue? y/n y

\*\*\*STUDENT DATABASE\*\*\*

1. Insert data

2. Search data

3. Read all data

Selection: 2

Insert roll no.: 1

Record found:

Name: bran

Marks: 72

Want to continue? y/n y

\*\*\*STUDENT DATABASE\*\*\*

1. Insert data

2. Search data

3. Read all data

Selection: 2

Insert roll no.: 6

Not found

Want to continue? y/n y

\*\*\*STUDENT DATABASE\*\*\*

1. Insert data

2. Search data

3. Read all data

Selection: 3

2 meera 46

1 bran 72

Want to continue? y/n n

# Assignment 29

***Objective***: Write a program to read marks of students from a binary file and show frequency of marks in every range (0 - 10, 10 - 20 etc. till 100).

***Code****:*

import pickle

marks = {}

for i in range(0, 101, 10):

    marks[i] = 0

with open(r'test\student.dat','rb') as f:

    while True:

        try:

            record = pickle.load(f)

            mark = record[2]

            category = (mark//10)\*10

            marks[category] += 1

        except:

            break

print('Summary')

for mark, f in marks.items():

    if mark == 100:

        print('100 Marks: ', f)

    elif f:

        print(f'From {mark} - {mark+10}: {f}')

***Output****:*

Summary

From 20 - 30: 1

From 50 - 60: 1

From 60 - 70: 2

From 70 - 80: 2

From 80 - 90: 1

From 90 - 100: 2

100 Marks:  1

# Assignment 30

***Objective***: Write Python application that fetches all records from employee table of ecorp database

***Code****:*

import mysql.connector as sq

db = sq.connect(host='localhost',user='root',passwd='root',database='ecorp')

cursor = db.cursor()

cursor.execute('SELECT \* FROM EMPLOYEE')

data = cursor.fetchall()

for row in data:

    print(row)

***Output****:*

(100, 'Elliot', 'Alderson', 250000)

(101, 'Darlene', 'Alderson', 110000)

(102, 'Angela', 'Moss', 75000)

(103, 'Tyrell', 'Wellick', 75000)

(104, 'Philip', 'Price', 55000)

(105, 'Sunil', 'Markesh', 69000)

(106, 'Dominique', 'DiPierro', 78000)

(107, 'Francis', 'Shaw', 70000)

(108, 'Shama', 'Biswas', 71000)

# Assignment 31

***Objective***: Write Python application that insert records in employee table of ecorp database. Take the record as input from user, as many as desired.

***Code****:*

import mysql.connector as sq

db = sq.connect(host='localhost',user='root',passwd='root',database='ecorp')

c = db.cursor()

ans = 'y'

c.execute('''CREATE TABLE IF NOT EXISTS emp(

        EID INT PRIMARY KEY,

        NAME VARCHAR(20),

        DEPT INT );''')

while ans == 'y':

    e\_id = int(input('\nEnter employee id: '))

    name = input('Enter employee name: ')

    dept = int(input('Enter employee dept no.: '))

    query = f"INSERT INTO emp values({e\_id},'{name}',{dept})"

    c.execute(query)

    ans = input('Record Added.\nContinue? y/n ')

    db.commit()

***Output****:*

Enter employee id: 1

Enter employee name: Elliot

Enter employee dept no.: 5

Record Added.

Continue? y/n y

Enter employee id: 2

Enter employee name: Meera

Enter employee dept no.: 3

Record Added.

Continue? y/n y

Enter employee id: 3

Enter employee name: Jojen

Enter employee dept no.: 3

Record Added.

Continue? y/n y

Enter employee id: 4

Enter employee name: Asha

Enter employee dept no.: 5

Record Added.

Continue? y/n n

***Table****:*



# Assignment 32

***Objective***: Write a python program that can update a record in employee table of ecorp database based on value of primary key given by the user.

***Code****:*

import mysql.connector as sq

db = sq.connect(host='localhost', user='root', password='root', database='ecorp')

c = db.cursor()

eid = int(input('Enter employee id to update: '))

c.execute(f'SELECT \* FROM emp WHERE eid = {eid}')

row = c.fetchone()

def update\_column(column, new\_val):

    query = f'update emp set {column} = {new\_val} where eid = {eid}'

    c.execute(query)

    db.commit()

if row:

    print(f'One record found:')

    print(f'ID: {eid}\nName: {row[1]}\nDept: {row[2]}\nSalary: {row[3]}\n')

    column = input('Enter column to update: ')

    new\_val = int(input(f'Enter new value: '))

    while True:

        try:

            update\_column(column, new\_val)

            break

        except:

            column = input('Invalid column. Enter column to update: ')

            update\_column(column, new\_val)

    print('Record updated.')

else:

    print('No such record exists.')

***Output****:*

Enter employee id to update: 3

One record found:

ID: 3

Name: Jojen

Dept: 3

Salary: 3500

Enter column to update: dept

Enter new value: 4

Record updated.

>>>

Enter employee id to update: 1

One record found:

ID: 1

Name: Elliot

Dept: 5

Salary: 4000

Enter column to update: sal

Enter new value: 4800

Invalid column. Enter column to update: salary

Record updated.

***Table****:*



# Assignment 33

***Objective***: Write Python application that provides the user the choice, either to add a column or modify an exisiting column emp table of ecorp database. Take the required input from user.

***Code****:*

import mysql.connector as sq

db = sq.connect(host='localhost', user='root', password='root', database='ecorp')

c = db.cursor()

ch = int(input('Enter whether to \n1. Add Column or \n2. Change column\nChoice : '))

if ch == 1:

    table = input("enter table name : ")

    name = input("enter column name : ")

    typ = input("enter its data type : ")

    length = int(input("enter column\'s length : "))

    query = f"alter table {table} add {name} {typ}({length})"

    c.execute(query)

elif ch == 2:

    table=input("enter table name : ")

    cname=input("enter column name : ")

    typ = input("enter its data type : ")

    length = input("enter column\'s length : ")

    const=input("enter constraint name(if any) : ")

    query = f"alter table {table} change {cname} {cname} {typ}({length}) {const}"

    c.execute(query)

print('Table updated.')

db.commit()

***Output****:*

Enter whether to

1. Add Column or

2. Change column

Choice : 1

enter table name : emp

enter column name : job

enter its data type : varchar

enter column's length : 20

Table updated.

>>>

Enter whether to

1. Add Column or

2. Change column

Choice : 2

enter table name : emp

enter column name : name

enter its data type : varchar

enter column's length : 15

enter constraint name(if any) : unique

Table updated.

***Table Before****:*



***Table After****:*



# Assignment 34

***Objective***: Write a python application that fetches the population of every continent from the country table in world databse and lists them in decreasing order. Use SQL group statements to do the same.

***Code****:*

import mysql.connector as sq

db = sq.connect(host='localhost', user='root', password='root', database='world')

c = db.cursor()

table = 'country'

col1 = 'continent'

col2 = 'population'

query = f'select {col1}, sum({col2}) from {table} group by {col1} order by sum({col2}) desc'

c.execute(query)

data = c.fetchall()

print('Continent\tPopulation')

for row in data:

    print(f'{row[0]}:\t{row[1]}')

***Output****:*

Continent       Population

Asia:           3705025700

Africa:         784475000

Europe:         730074600

North America:  482993000

South America:  345780000

Oceania:        30401150

Antarctica:     0

# MySQL

*Table 1:* STUDENT

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Name** | **Stipend** | **Stream** | **AvgMark** | **Grade** | **Class** |
| 1 | Karan | 400 | Medical | 78.5 | B | 12B |
| 2 | Divakar | 450 | Commerce | 89.2 | A | 11C |
| 3 | Divya | 300 | Commerce | 68.6 | C | 12C |
| 4 | Arun | 350 | Humanities | 73.1 | B | 12C |
| 5 | Sabina | 500 | Nonmedical | 90.6 | A | 11A |
| 6 | John | 400 | Medical | 75.4 | B | 12B |
| 7 | Robert | 250 | Humanities | 64.4 | C | 11A |
| 8 | Rubina | 450 | Nonmedical | 88.5 | A | 12A |
| 9 | Vikas | 500 | Nonmedical | 92.0 | A | 12A |
| 10 | Mohan | 300 | Commerce | 67.5 | C | 12C |

# Assignment 35

***Objective***: Solve problems related to fetching data from the table

1. **List all names of those students who are in class 12 sorted by Stipend.**

**Answer:**

Select name, stipend, class FROM student

Where class like ‘12%’

Order by stipend;

**Output:**



1. **List the Name and Stream of the students whose Grade is A.**

**Answer:**

select name, stream from student where grade = 'A';

**Output:**



**c) List the maximum and minimum mark in each stream**

**Answer:**

select stream,max(avgmark), min(avgMark) from student group by stream;

**Output:**



# Assignment 36

***Objective***: Solve problems related to updation and deletion of data in the table.

1. **Increase the stipend of medical students by 100.**

**Answer:**

update student set stipend = stipend + 100 where stream = 'Medical';

**Updated Table:**



1. **Delete student records where grade is 'C' or 'B'.**

**Answer:**

delete from student where grade in ('C', 'B');

**Updated Table:**



# Assignment 37

***Objective***: Solve problems related to altering table and its attributes.

1. **Make the name attribute not null.**

**Answer:**

ALTER TABLE student MODIFY name varchar(20) NOT NULL;

**Altered Table Description:**



1. **Delete the grade attribute from the table.**

**Answer:**

ALTER TABLE student DELETE COLUMN grade;

**Altered Table Description:**



1. **Rename ‘AvgMark’ attribute to ‘Mark’.**

**Answer:**

ALTER TABLE student CHANGE AvgMark Mark float(3);

**Altered Table Description:**



Table: *emp*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **emp\_id** | **name** | **job** | **sales** | **salary** | **deptno** |
| 100 | Elliot | manager | 8902 | 78000 | 2 |
| 101 | Darlene | salesman | 8698 | 60000 | 3 |
| 102 | Angela | salesman | 8698 | 75000 | 3 |
| 103 | Tyrell | clerk | 8839 | 63000 | 2 |
| 104 | Philip | salesman | 7956 | 55000 | 3 |
| 105 | Sunil | manager | 8523 | 69000 | 3 |
| 106 | Dominique | president | NULL | 250000 | 1 |
| 107 | Francis | salesman | 8345 | 70000 | 3 |
| 108 | Shama | manager | 8965 | 71000 | 1 |
| 109 | Alderson | clerk | 7942 | 63000 | 2 |
| 110 | Arya | analyst | 8156 | 70000 | 2 |
| 111 | Anoop | clerk | 8356 | 55000 | 3 |
| 112 | Fahad | analyst | 8880 | 70000 | 3 |
| 113 | Bina | clerk | 9010 | 58000 | 1 |

Table: *depts*

|  |  |  |
| --- | --- | --- |
| **deptno** | **dept\_name** | **mgr\_id** |
| 1 | Corporate | 100 |
| 2 | Scranton | 102 |
| 3 | Stamford | 106 |

# Assignment 38

***Objective***: Solve problems related to fetching data from the table.

1. **Display only the jobs with maximum salary greater than 70000.**

**Answer:**

SELECT job FROM emp

GROUP BY job HAVING MAX(salary) > 70000;

**Output:**



1. **Show the average salary of all departments with less than 3 employees for a job.**

**Answer:**

SELECT deptno, job, avg(salary) FROM emp

GROUP BY deptno, job

HAVING count(job) < 3;

**Output:**



1. **List the numbers of employees having 'Manager' as the job in every department.**

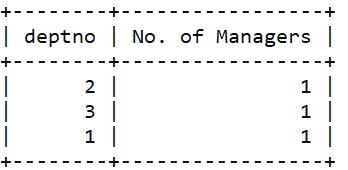
**Answer:**

SELECT deptno, count(\*) 'No. of Managers' FROM emp

WHERE job = 'manager'

GROUP BY deptno;

**Output:**



# Assignment 39

***Objective***: Solve problems related combination of values from two different table connected using a foreign key.

1. **Identify the foreign key in table emp.**

**Answer:**

deptno

1. **List employee name and their corresponding department name sorted by department name and then employee name.**

**Answer:**

SELECT name, dept\_name

FROM emp, depts

WHERE emp.deptno = depts.deptno

ORDER BY dept\_name, name;

**Output:**



1. **List the name of the employee with maximum sales in each department along with the department name and sales**

**Answer:**

SELECT emp.deptno, dept\_name, name 'Employee with max sales', sales

FROM emp, depts

WHERE sales = (SELECT MAX(sales) FROM emp WHERE emp.deptno = depts.deptno)

GROUP BY emp.deptno;

**Output:**



1. **List the number of unique jobs in each department.**

**Answer:**

SELECT dept\_name, count(DISTINCT job) 'No. of unique jobs'

FROM emp, depts

GROUP BY emp.deptno;

**Output:**



# Assignment 40

***Objective***: Solve problems related to altering table and its attributes.

1. **Add attribute branch to the job table with default value of 'Corporate'.**

**Answer:**

ALTER TABLE emp ADD branch varchar(20) DEFAULT 'Corporate';

**Output:**



1. **Change the name of the table to employees.**

**Answer:**

RENAME TABLE emp TO employees;

**Output:**

