|  |  |
| --- | --- |
|  |  |

**Faculty of Technology and Engineering**

**U & P U. Patel Department of Computer Engineering**

Date: 01 / 12 / 2021

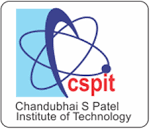
**Practical List**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Academic Year | : | 2021-22 | Semester | : | 4 |
| Course code | : | CE259 | Course name | : | Programming in Python |

**Note: Practical List is for Students. We need to cover concept require to implement respective practical**

|  |  |
| --- | --- |
| **Sr. No.** | **Aim** |
|  | Installation & Configuration of Python(**3.6 or 3.7**) and Virtual Environment. Along with its all major editors, IDLE, Pycharm, Anaconda, Jupyter, Interpreter etc.  **Note: Do not install the latest version of python due to some backward compatibility issues.**  **Please take screenshots of each point mentioned in the assignment and upload .pdf file.** |
|  | Pycharm: |
|  |  |
|  | Anaconda: |
|  | Python.exe |

20CS015 Dev Desai



Faculty of Technology and Engineering

# U & P U. Patel Department of Computer Engineering

Date: 10 / 12 / 2021

# Practical List

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Academic Year | : | 2021-22 | Semester | : | 4 |
| Course code | : | CE259 | Course name | : | Programming in Python |

## AIM:

*# Write a Python script to check whether a given key already exists in a dictionary.*

*# Developed By: Dev Desai(20CS015)*

*# https://github.com/devdesai0602/Programming-in-Python*

**def** checkKey(dict, key):

**if** key **in** dict:

print("Present, ", end =" ") print("value =", dict[key])

**else**:

print("Not present")

*# Driver Code*

dict = {'a': 100, 'b':200, 'c':300}

key = 'b' checkKey(dict, key)

key = 'w' checkKey(dict, key)

## OUTPUT:

Present, value = 200 Not present

## AIM:

*# Write a Python script to merge two Python dictionaries. # Developed By: Dev Desai(20CS015)*

*# https://github.com/devdesai0602/Programming-in-Python*

d1 = {'a': 100, 'b': 200}

d2 = {'x': 300, 'y': 200}

d = d1.copy() *#print(Merge(d1,d2))*

d.update(d2) print(d)

## OUTPUT:

{'a': 100, 'b': 200, 'x': 300, 'y': 200}

## AIM:

*# Write a Python program to sum all the items in a dictionary.*

*# my\_dict = {'data1':100,'data2':-54,'data3':247} # print(sum(my\_dict.values()))*

*# Developed By: Dev Desai(20CS015)*

*# https://github.com/devdesai0602/Programming-in-Python*

my\_dict = {'A': 100, 'B': 200, 'C': 300}

sum = 0

**for** i **in** my\_dict.values(): sum = sum + i

print(sum)

## OUTPUT:

600

## AIM:

*# Write a Python script to add a key to a dictionary. # Developed By: Dev Desai(20CS15)*

*# https://github.com/devdesai0602/Programming-in-Python*

d = {0:10, 1:20}

print(d) d.update({2:30}) print(d)

## OUTPUT:

{0: 10, 1: 20}

{0: 10, 1: 20, 2: 30}

## AIM:

*# Write a Python script to concatenate following dictionaries to create a new one.*

*# dic1={1:10, 2:20}*

*# dic2={3:30, 4:40} # dic3={5:50,6:60}*

*# Developed By: Dev Desai(20CS015)*

*# https://github.com/devdesai0602/Programming-in-Python*

dic1={1:10, 2:20}

dic2={3:30, 4:40}

dic3={5:50,6:60}

dic4 = {}

**for** d **in** (dic1, dic2, dic3): dic4.update(d) print(dic4)

## OUTPUT:

{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

## AIM:

*# Write a Python program to create a tuple with different data types. # Developed By: Dev Desai(20CS015)*

*# https://github.com/devdesai0602/Programming-in-Python*

tuplex = ("tuple", False, 3.2, 1) print(tuplex)

## OUTPUT:

('tuple', False, 3.2, 1)

## AIM:

*# Write a Python program to create a tuple with numbers and print one item.*

*# Developed By: Dev Desai(20CS015)*

*# https://github.com/devdesai0602/Programming-in-Python*

fruits=('Banana','Apple','Mango','Orange') print(fruits)

print(fruits[2])

('Banana', 'Apple', 'Mango', 'Orange')

## OUTPUT:

Mango

## AIM:

*# Write a Python program to add an item in a tuple. # Developed By: Dev Desai(20CS015)*

*# https://github.com/devdesai0602/Programming-in-Python*

*#create a tuple*

tuplex = (4, 6, 2, 8, 3, 1) print(tuplex)

*#tuples are immutable, so you can not add new elements*

*#using merge of tuples with the + operator you can add an element and it will create a new tuple*

tuplex = tuplex + (9,)

print(tuplex)

*#adding items in a specific index*

tuplex = tuplex[:5] + (15, 20, 25) + tuplex[:5] print(tuplex)

## OUTPUT:

|  |  |  |
| --- | --- | --- |
| (4, 6, 2, 8, | 3, | 1) |
| (4, 6, 2, 8, | 3, | 1, 9) |
| (4, 6, 2, 8, | 3, | 15, 20, 25, 4, 6, 2, 8, 3) |

***AIM:***

*# Write a Python program to convert a tuple to a string. # Developed By: Dev Desai(20CS015)*

*# https://github.com/devdesai0602/Programming-in-Python*

tup = ('D', 'e', 'v', ' ', 'D', 'e', 's', 'a', 'i')

str = ''.join(tup) print(str)

## OUTPUT:

Dev Desai

## AIM:

*# Write a Python program to find the length of a tuple. # Developed By: Dev Desai(20CS015)*

*# https://github.com/devdesai0602/Programming-in-Python*

tuplex = tuple("Dev Desai") print(tuplex) print(len(tuplex))

## OUTPUT:

('D', 'e', 'v', ' ', 'D', 'e', 's', 'a', 'i') 9

## AIM:

*# Write a Python program to add member(s) in a set and clear a set # Developed By: Dev Desai(20CS015)*

*# https://github.com/devdesai0602/Programming-in-Python*

sports={'Cricket','Football','Hockey'} print(sports)

sports.add('Kabaddi')

print(sports) sports.update(['Volleyball','Baseball','Tennis']) print(sports)

print(sports.clear())

## OUTPUT:

{'Hockey', 'Football', 'Cricket'}

{'Hockey', 'Football', 'Cricket', 'Kabaddi'}

{'Hockey', 'Football', 'Cricket', 'Baseball', 'Kabaddi', 'Tennis', 'Volleyball'}

None

## AIM:

*# Write a Python program to remove an item from a set if it is present in the set.*

*# Developed By: Dev Desai(20CS015)*

*# https://github.com/devdesai0602/Programming-in-Python*

fruits={'Mango','Apple','Orange','Grapes','Banana'} print(fruits)

fruits.remove('Banana') print(fruits)

## OUTPUT:

{'Grapes', 'Banana', 'Orange', 'Mango', 'Apple'}

{'Grapes', 'Orange', 'Mango', 'Apple'}

## AIM:

*# Write a Python program to create an intersection, Union, difference of sets.*

*# Developed By: Dev Desai(20CS015)*

*# https://github.com/devdesai0602/Programming-in-Python*

A = {0, 2, 4, 6, 8};

B = {1, 2, 3, 4, 5};

print('Intersection :', A.intersection(B)) print('Union :', A.union(B)) print('Difference :', A.difference(B))

## OUTPUT:

Intersection : {2, 4}

Union : {0, 1, 2, 3, 4, 5, 6, 8}

Difference : {0, 8, 6}

## AIM:

*# Write a Python program to find maximum and the minimum value in a set.*

*# Developed By: Dev Desai(20CS015)*

*# https://github.com/devdesai0602/Programming-in-Python*

numbers = {51, 110, 13, 5, 2, 200}

print(numbers)

print('Maximum :', max(numbers)) print('Minimum :', min(numbers))

## OUTPUT:

{2, 51, 5, 200, 13, 110}

Maximum : 200

Minimum : 2

## AIM:

*# Write a Python program to find the most common elements and their counts from list, tuple, dictionary.*

*# Developed By: Dev Desai(20CS015)*

*# https://github.com/devdesai0602/Programming-in-Python*

*#LIST*

print("For List") fruits=['apple','banana','orange','mango','mango','grapes','mango'] res = max(set(fruits), key = fruits.count) number=fruits.count(res)

print("Element with highest frequency :",res) print("Number of times element repeated :", number) print("\n")

*#TUPLE*

print("For Tuple") fruits\_tuple=('apple','banana','orange','mango','orange','grapes','ora nge')

res = max(set(fruits\_tuple), key = fruits\_tuple.count) number=fruits\_tuple.count(res)

print("Element with highest frequency :",res) print("Number of times element repeated :", number) print("\n")

*#DICTIONARY*

print("For Dictionary") student={

'name1':'ABC',

'name2':'PQR',

'name3':'ABC',

'name4':'XYZ',

'name5':'LMN',

'name6':'IJK',

}

tp=tuple(student.values()) count=0

element=tp[0]

**for** i **in** tp:

ctr=tp.count(i)

**if**(ctr>count): count=ctr element=i

print("Element with highest frequency :",element)

## OUTPUT:

For List

Element with highest frequency : mango Number of times element repeated : 3

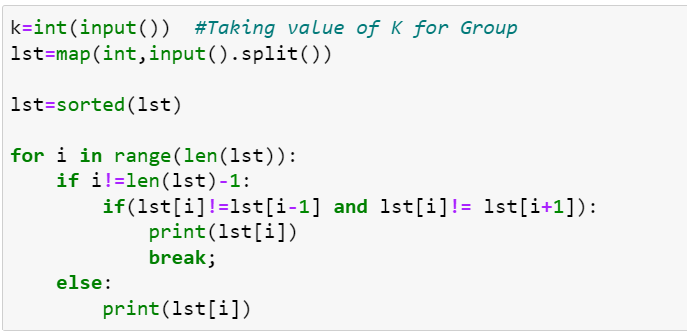
For Tuple

Element with highest frequency : orange Number of times element repeated : 3

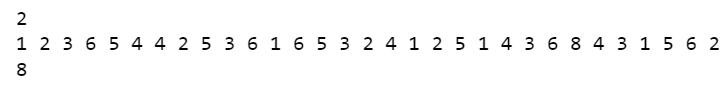
For Dictionary

Element with highest frequency : ABC

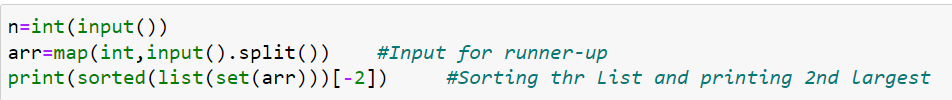
Code:



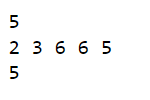
Output:



Code:



Output:



|  |  |
| --- | --- |
|  |  |

Faculty of Technology and Engineering

# U & P U. Patel Department of Computer Engineering

# Practical List

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Academic Year | : | 2021-22 | Semester | : | 4 |
| Course code | : | CE259 | Course name | : | Programming in Python |

**Note: Practical List is for Students. We need to cover concept require to implement respective practical**

* ***AIM:***

You are given a string and your task is to swap cases. In other words, convert all lowercase letters to uppercase letters and vice versa.

Sample Input: HackerRank.com presents "Pythonist 2". Sample Output: hACKERrANK.COM PRESENTS "pYTHONIST 2".

* ***CODE & OUTPUT:***





* ***CONCLUSION:***

I have learnt how to use function of swapping the case of the String.

The string swapcase() method converts all uppercase characters to lowercase and vice versa of the given string, and returns it.

## Syntax:

string\_name.swapcase()

Here string\_name is the string whose cases are to be swapped.

## Parameter:

The swapcase() method does not takes any parameter.

## Return value:

The swapcase() method returns a string with all the case s changed.

|  |  |
| --- | --- |
|  |  |

Faculty of Technology and Engineering

## U & P U. Patel Department of Computer Engineering

Date: 11 / 3 / 2022

## Practical List

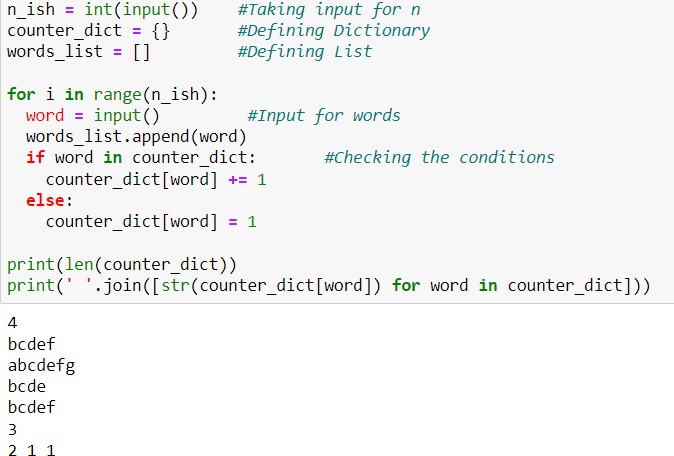
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Academic Year | : | 2021-22 | Semester | : | 4 |
| Course code | : | CE259 | Course name | : | Programming in Python |

**Note: Practical List is for Students. We need to cover concept require to implement respective practical**

* ***AIM:***

You are given n words. Some words may repeat. For each word, output its number of occurrences. The output order should correspond with the input order of appearance of the word. See the sample input/output for clarification.

# * CODE & OUTPUT:



***Software used:*** Anaconda

# * CONCLUSION:

[Python Dictionary](https://www.geeksforgeeks.org/python-dictionary/) is like a map that is used to store data in the form of a key:value pair. Python provides various in-built functions to deal with dictionaries.

Lists are just like dynamically sized arrays, declared in other languages (vector in C++ and ArrayList in Java). Lists need not be homogeneous always which makes it the most powerful tool in [Python](https://www.geeksforgeeks.org/python-programming-language/). A single list may contain DataTypes like Integers, Strings, as well as Objects. Lists are mutable, and hence, they can be altered even after their creation.

|  |  |
| --- | --- |
|  |  |

Faculty of Technology and Engineering

## U & P U. Patel Department of Computer Engineering

Date: 10 / 3 / 2022

## Practical List

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Academic Year | : | 2021-22 | Semester | : | 4 |
| Course code | : | CE259 | Course name | : | Programming in Python |

**Note: Practical List is for Students. We need to cover concept require to implement respective practical**

* ***AIM:***

Given a string, you need to tell if it is a lapindrome.

# * CODE & OUTPUT:

***Software used:*** Anaconda

# * CONCLUSION:

Lapindrome is defined as a string which when split in the middle, gives two halves having the same characters and same frequency of each character. If there are odd number of characters in the string, we ignore the middle character and check for lapindrome. For example gaga is a lapindrome, since the two halves ga and ga have the same characters with same frequency. Also, abccab, rotor and xyzxy are a few examples of lapindromes. Note that abbaab is NOT a lapindrome. The two halves contain the same characters but their frequencies do not match.

|  |  |
| --- | --- |
|  |  |

Faculty of Technology and Engineering

## U & P U. Patel Department of Computer Engineering

Date: 15 / 03 / 2022

## Practical List

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Academic Year | : | 2021-22 | Semester | : | 4 |
| Course code | : | CE259 | Course name | : | Programming in Python |

**Note: Practical List is for Students. We need to cover concept require to implement respective practical**

# * AIM:

Write a Program in Python to implement a Stack Data Structure using Class and Objects, with push, pop, and traversal method.

# * CODE & OUTPUT:

# Aim: Write a Program in Python to implement a Stack Data Structure using Class and Objects,

# With push, pop, and traversal method. # Stack implementation in python

# Creating a stack def create stack():

stack = [] return stack

# Creating an empty stack def check\_empty(stack):

return len(stack) == 0

# Adding items into the stack def push(stack, item):

stack.append(item) print("pushed item: " + item)

# Removing an element from the stack

def pop(stack):

if (check\_empty(stack)): return "stack is empty"

return stack.pop()

stack = create\_stack() push(stack, str(1)) push(stack, str(2)) push(stack, str(3)) push(stack, str(4))

print("popped item: " + pop(stack))

print("stack after popping an element: " + str(stack))



***Software used:*** VS CODE

# * CONCLUSION:

A stack is a linear data structure that stores items in a Last-In/First-Out (LIFO) or First-In/Last-Out (FILO) manner. In stack, a new element is added at one end and an element is removed from that end only. The insert and delete operations are often called push and pop.

|  |  |
| --- | --- |
|  |  |

Faculty of Technology and Engineering

## U & P U. Patel Department of Computer Engineering

Date: 15 / 03 / 2022

## Practical List

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Academic Year | : | 2021-22 | Semester | : | 4 |
| Course code | : | CE259 | Course name | : | Programming in Python |

**Note: Practical List is for Students. We need to cover concept require to implement respective practical**

# * AIM:

Consider an example of declaring the examination result. Design three classes: Student, Exam, and Result. The Student class has data members such as those representing rollNumber, Name, etc. Create the class Exam by inheriting Student class. The Exam class adds fields representing the marks scored in six subjects.

Derive Result from the Exam class, and it has its own fields such as total\_marks. Write an interactive program to model this relationship.

# * CODE:

# Consider an example of declaring the examination result. Design three classes: Student, Exam,

# and Result.

# The Student class has data members such as those representing rollNumber, Name, etc.

# Create the class Exam by inheriting Student class. class Student:

def init (self, rollno, name, age): self.rollno = rollno

self.name = name self.age = age

# The Exam class adds fields representing the marks scored in three subjects.

class Exam(Student):

marks = {

"Physics": 0,

"Chemistry": 0,

"Maths": 0,

}

def init (self, rollno, name, age, marks): super(). init (rollno, name, age) self.marks = marks

# Derive Result from the Exam class, and it has its own fields such as total\_marks.

class Result(Exam):

def init (self, rollno, name, age, marks): super(). init (rollno, name, age, marks) self.total\_marks = sum(self.marks.values())

def display(self):

print("Roll Number:", self.rollno) print("Name:", self.name) print("Age:", self.age) print("Marks:", self.marks) print("Total Marks:", self.total\_marks)

# Write an interactive program to model this relationship. if name == " main ":

rollno = int(input("Enter Roll Number: ")) name = input("Enter Name: ")

age = int(input("Enter Age: ")) marks = {

"Physics": int(input("Enter Physics Marks: ")), "Chemistry": int(input("Enter Chemistry Marks: ")), "Maths": int(input("Enter Maths Marks: ")),

}

result = Result(rollno, name, age, marks) result.display()

***Software used:*** VS CODE

# * CONCLUSION:

How the inheritance, constructor and function definition works in python.

|  |  |
| --- | --- |
|  |  |

Faculty of Technology and Engineering

## U & P U. Patel Department of Computer Engineering

Date: 15 / 03 / 2022

## Practical List

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Academic Year | : | 2021-22 | Semester | : | 4 |
| Course code | : | CE259 | Course name | : | Programming in Python |

**Note: Practical List is for Students. We need to cover concept require to implement respective practical**

**Aim:-Generate PDF of 3rd sem.**

**Code:**

**from fpdf import FPDF**

**pdf = FPDF()**

**pdf.add\_page()**

**pdf.set\_font("Arial", size = 10)**

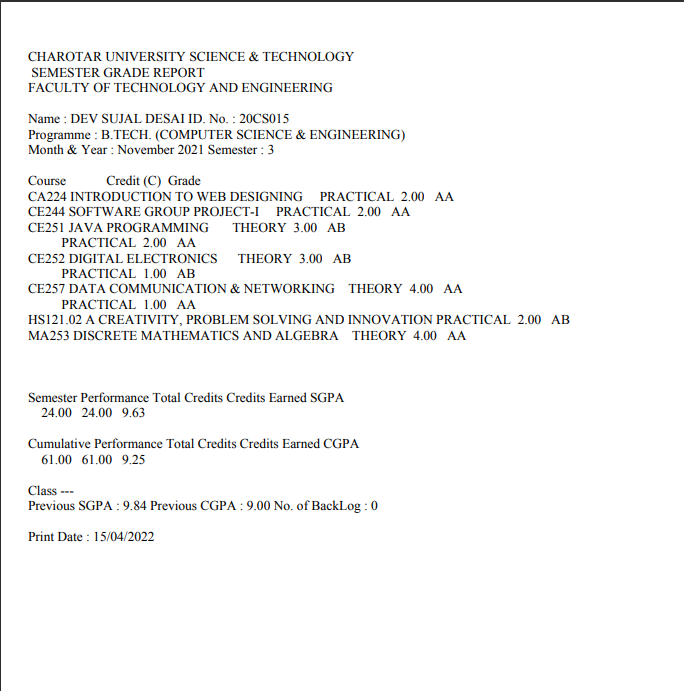
**f = open(r"C:\Users\dhruv\OneDrive\Desktop\DHRUV\4th SEM\PIP\PRACTICALS\marksheet.txt")**

**for i in f:**

**pdf.cell(200,10,txt=i,ln=1)**

**pdf.output("mark-sheet.pdf")**

**Output:**

****