

## Practical - 6

**AIM: 1)WAP to create function that takes a number as a parameter and check the number is prime or not.**

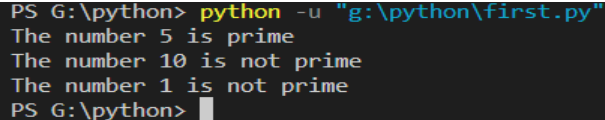
```
def fun(a):  
    if a>=2:  
        for i in range(2,a):  
            if a%i==0:  
                print("The number",a,"is not prime")  
                return  
        print("The number",a,"is prime")  
    else:  
        print("The number",a,"is not prime")
```

fun(5)

fun(10)

fun(1)

### OUTPUT:

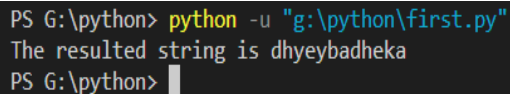


```
PS G:\python> python -u "g:\python\first.py"  
The number 5 is prime  
The number 10 is not prime  
The number 1 is not prime  
PS G:\python>
```

**AIM: 2)WAP to add two string using a function.**

```
def fun(a,b):  
    return a+b  
c=fun("dhyey","badheka")  
print("The resulted string is",c)
```

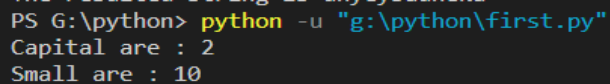
### OUTPUT:



```
PS G:\python> python -u "g:\python\first.py"  
The resulted string is dhyeybadheka  
PS G:\python>
```

**AIM: 3)WAP to calculate total uppercase and lowercase of given string.**

```
def fun(a):  
    cap=0  
    small=0  
    for i in a:  
        if ord(i)>=65 and ord(i)<=91:  
            cap+=1  
        if ord(i)>=97 and ord(i)<=123:  
            small+=1  
    print("Capital are : "+ str(cap))  
    print("Small are : "+ str(small))  
fun("DhyeyBadheka")
```

**OUTPUT:**

```
PS G:\python> python -u "g:\python\first.py"  
Capital are : 2  
Small are : 10
```

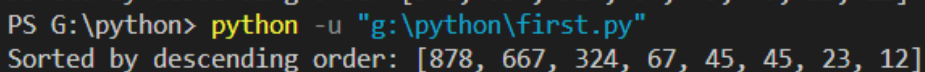
**AIM: 4)WAP to arrange given numbers in descending orders import module of ascending order.**

**first.py**

```
import sortingmod  
a=[324,45,667,23,12,67,878,45]  
b=sortingmod.sortele(a)  
b.sort(reverse=True)  
print("Sorted by descending order:",b)
```

**sortingmod.py**

```
def sortele(a):  
    a=sorted(a)  
    return a
```

**OUTPUT:**

```
PS G:\python> python -u "g:\python\first.py"  
Sorted by descending order: [878, 667, 324, 67, 45, 45, 23, 12]
```

## Practical - 7

**AIM: 1) WAP to add complex number using class and object.**

```
import math

class add_comp:

    def __init__(self,a,b):
        self.obj=complex(a,b)

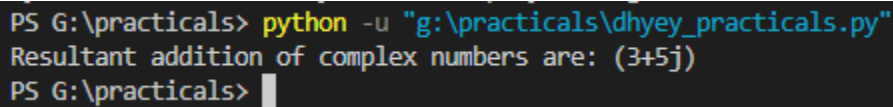
    def addc(self,c1):
        r=self.obj+c1.obj

        return r

a=add_comp(1,2)
b=add_comp(2,3)
c=a.addc(b)

print("Resultant addition of complex numbers are:",c)
```

**OUTPUT:**



```
PS G:\practicals> python -u "g:\practicals\dhyey_practicals.py"
Resultant addition of complex numbers are: (3+5j)
PS G:\practicals>
```

**AIM: 2) WAP to check string is palindrome or not using class**

```
class palin:

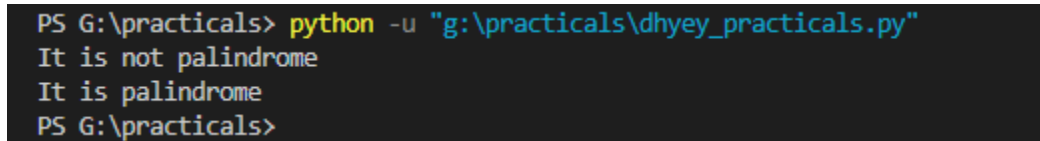
    def checkpalin(a):
        counter=0

        a=str(a)

        for i in range(int(len(a)/2)):
            if a[i]==a[len(a)-i-1]:
                counter+=1

        if counter==int(len(a)/2):
            print("It is palindrome")
        else:
            print("It is not palindrome")
```

```
a="dhyey"  
b="madam"  
palin.checkpalin(a)  
palin.checkpalin(b)
```

**OUTPUT:**

```
PS G:\practicals> python -u "g:\practicals\dhyey_practicals.py"  
It is not palindrome  
It is palindrome  
PS G:\practicals>
```

**AIM: 3) WAP multiply the positive numbers using multiple inheritance.**

```
class class1:  
    def mul(self,a):  
        global multi  
        multi=1  
        for i in a:  
            if i>0:  
                multi*=i  
        return multi  
class class2(class1):  
    def mul(self,a):  
        pass  
class class3(class2,class1):  
    def mul(self,a):  
        pass  
d=class1()  
a=[1,2,3,4,-5,6,-7]  
e=d.mul(a)  
print(e)
```

**OUTPUT:**

```
PS G:\practicals> python -u "g:\practicals\dhyey_practicals.py"
144
PS G:\practicals>
```

**AIM: 3) WAP to create employee salary system for PF, TDS and 80C.**

```
class emp:
```

```
    def __init__(self,obj):
```

```
        self.salary=obj
```

```
        self.pf=self.salary*0.6
```

```
        self.tds=self.salary*0.4
```

```
        self.cess=self.salary*0.3
```

```
        self.total=self.salary+self.pf+self.tds-self.cess
```

```
    def printdet(self):
```

```
        print("Salary is",self.salary)
```

```
        print("PF is",self.pf)
```

```
        print("TDS is",self.tds)
```

```
        print("Cess is",self.cess)
```

```
        print("Total salary is",self.total)
```

```
a=emp(1000)
```

```
a.printdet()
```

**OUTPUT:**

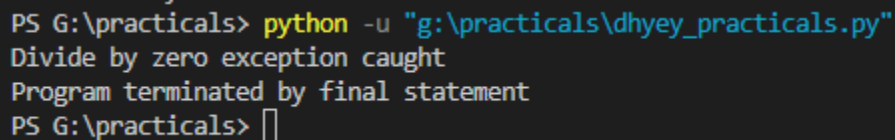
```
PS G:\practicals> python -u "g:\practicals\dhyey_practicals.py"
Salary is 1000
PF is 600.0
TDS is 400.0
Cess is 300.0
Total salary is 1700.0
PS G:\practicals>
```

## Practical - 8

**AIM: 1) WAP to perform exception handling for Divide by zero Exception.**

```
a=10
b=0
try:
    c=a/b
except ZeroDivisionError:
    print("Divide by zero exception caught")
else:
    print("The division is",c)
finally:
    print("Program terminated by final statement")
```

### OUTPUT:

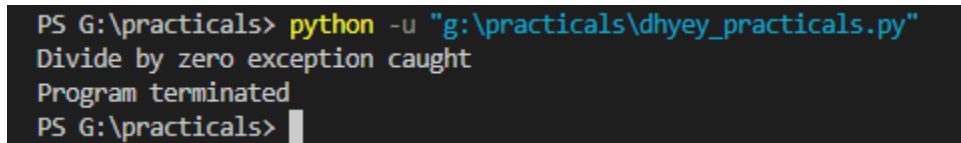


```
PS G:\practicals> python -u "g:\practicals\dhyey_practicals.py"
Divide by zero exception caught
Program terminated by final statement
PS G:\practicals> █
```

**AIM: 2) WAP to perform exception handling operation.**

```
a=10
b=0
try:
    c=a/b
    d="Dhyey"
    print(d[5])
except ZeroDivisionError:
    print("Divide by zero exception caught")
except IndexError:
    print("Index out of bound exception caught")
except:
```

```
print("Exception other than Zerodivision and index out of bound ")
else:
    print(c)
    print(d)
finally:
    print("Program terminated")
```

**OUTPUT:**

```
PS G:\practicals> python -u "g:\practicals\dhyey_practicals.py"
Divide by zero exception caught
Program terminated
PS G:\practicals>
```

**AIM: 3) WAP to create user defined Exception.**

```
class userdef(Exception):
    pass
a=7
try:
    if a>5:
        b=7
        raise userdef(a,b)
    else:
        c=10/3
except userdef:
    a*=2
    print(f"User Defined exception called with following paramteres: {a} {b}")
except Exception as e:
    print("Exception called :-- " + str(e))
else:
    print("Else called")
finally:
```

```
print("This will be always called")
```

**OUTPUT:**

```
PS G:\practicals> python -u "g:\practicals\dhyey_practicals.py"
User Defined exception called with following paramteres: 14 7
This will be always called
PS G:\practicals>
```

**AIM: 4) WAP to show use of assert in exception.**

```
def valuesofa(a):
```

```
    assert a>20,"The value is not greater than 20"
```

```
    print("A is",a)
```

```
valuesofa(50)
```

```
valuesofa(5)
```

**OUTPUT:**

```
PS G:\practicals> python -u "g:\practicals\dhyey_practicals.py"
A is 50
Traceback (most recent call last):
  File "g:\practicals\dhyey_practicals.py", line 5, in <module>
    valuesofa(5)
  File "g:\practicals\dhyey_practicals.py", line 2, in valuesofa
    assert a>20,"The value is not greater than 20"
AssertionError: The value is not greater than 20
PS G:\practicals>
```



## Practical – 9

**AIM: 1) WAP to create a file and store given text and calculate total number of line and space and display it on shell.**

s="Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum."

```
f=open("prac9.txt","w")
f.write(s)
f.close()
f=open("prac9.txt","r")
stri=f.read()
f.close()
lisoffline=stri.split(".")
print("The number of lines are ",len(lisoffline)-1)
spacecount=stri.count(" ")
print("The number of spaces are ",spacecount)
print(f"The text is\n{stri}")
```

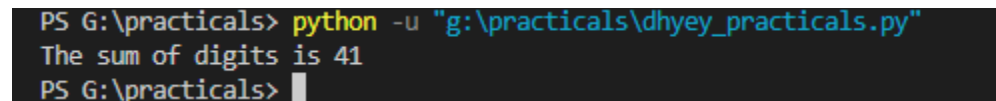
### OUTPUT:

```
PS G:\practicals> python -u "g:\practicals\dhyey_practicals.py"
The number of lines are 4
The number of spaces are 90
The text is
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard d
ummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen bo
ok. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchang
ed. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recentl
y with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.
PS G:\practicals>
```

**AIM: 2) WAP to store content of two file into one file and do addition of all the digit available in file.**

```
string1="This is Dhyey Badheka id is 19IT450 "
string2="This is Harsh Majithiya id is 19IT453 "
```

```
file=open("prac92.txt","w")
file=open("prac92.txt","a")
file.write(string1)
file.write(string2)
file.close()
file=open("prac92.txt","r")
stri=file.read()
li=stri.split(" ")
sum=0
for i in li:
    if i.isalnum():
        for j in i:
            if j.isdigit():
                sum+=int(j)
print("The sum of digits is",sum)
```

**OUTPUT:**

```
PS G:\practicals> python -u "g:\practicals\dhyey_practicals.py"
The sum of digits is 41
PS G:\practicals>
```

**AIM: 3) WAP to create histogram of all the letter available in file.**

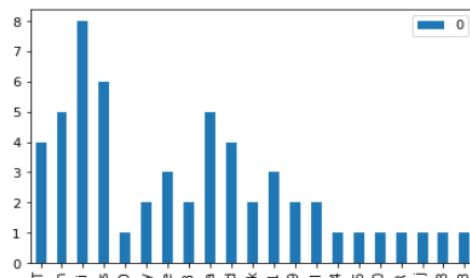
```
import pandas
import matplotlib
string1="This is Dhyey Badheka id is 19IT450"
string2="This is Raj Badheka id is 19IT813"
file=open("prac92.txt","w")
file=open("prac92.txt","a")
file.write(string1)
file.write(string2)
file.close()
```

```
file=open("prac92.txt","r")
stri=file.read()
dic={}
for i in stri:
    for j in i:
        if j == " ":
            continue
        elif j not in dic.keys():
            dic[j]=1
        else:
            dic[j]+=1
print(dic)
df = pandas.DataFrame.from_dict(dic, orient='index')
df.plot(kind='bar')
```

## OUTPUT:

```
{'T': 4, 'h': 5, 'i': 8, 's': 6, 'D': 1, 'y': 2, 'e': 3, 'B': 2, 'a': 5, 'd': 4, 'k': 2, 'l': 3, '9': 2, 'I': 2, '4': 1, '5': 1, '0': 1, 'R': 1, 'j': 1, '8': 1, '3': 1}
```

Out[1]: <AxesSubplot:>



## Practical – 10

**AIM: 1) WAP to find out following mobile number and person using regular expression. (111-111-1111, Mr. X, Mrs. Y, Mr ABC)**

```
import re
```

```
str="Hello Mr.Dhyey your mobile number is 7984528154 and Shree.ABC will be your secretary  
whose number is 9249252360 and reference number is 1334567234562"
```

```
name=re.compile("(Mr.|Mrs.|Smt.|Shree.)([a-zA-Z]+)")
```

```
match=re.findall(name,str)
```

```
print("Names are:")
```

```
for i in match:
```

```
    print(i[1])
```

```
num=re.compile("\s([0-9]{10})\s")
```

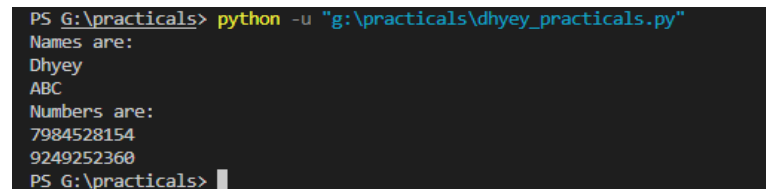
```
match=re.findall(num,str)
```

```
print("Numbers are:")
```

```
for i in match:
```

```
    print(i)
```

### OUTPUT:



```
PS G:\practicals> python -u "g:\practicals\dhyey_practicals.py"
Names are:
Dhyey
ABC
Numbers are:
7984528154
9249252360
PS G:\practicals> |
```

**AIM: 2) WAP to replace pincode in a given file using regular expression.**

```
import re
```

```
file=open("Pincode program.txt","r")
```

```
str=file.read()
```

```
print(str)
```

```
file.close()
```

```
regex=re.compile("\s[0-9]{6}\s")
```

```
match=re.findall(regex,str)
```

```
myPincode=eval(input("Enter pincode of your choice"))
```

```
myPincode=" "+myPincode+" "  
str=str.replace(match[0],myPincode)  
file=open("Pincode program.txt","w")  
file.write(str)  
file.close()  
file=open("Pincode program.txt","r")  
str=file.read()  
print(str)  
file.close()
```

### OUTPUT:

```
Hello I am Dhyey Badheka and am resident of Bhavnagar 342456 and currently in Ananad  
Enter pincode of your choice"388120"  
Hello I am Dhyey Badheka and am resident of Bhavnagar 388120 and currently in Ananad
```

### AIM: 3) WAP to find out name starts with letter R or S and ends with H or J.

import re

```
names="Dhyey Rohil Shah ddfgr dfbsdjfbfrfke snfajkdwakf Shamsheerj Raunak Rahil Badheka  
Nishith Harsh endwithj bdsjdbwj "
```

```
list=names.split(" ")
```

```
regex=re.compile("^(R|S)|(H|J)$", re.IGNORECASE)
```

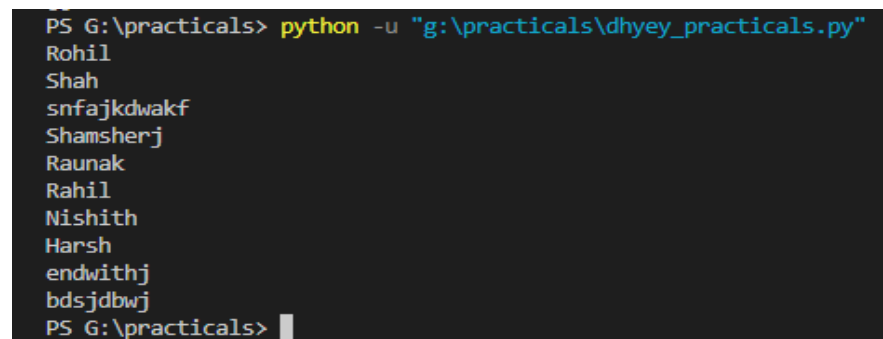
```
for i in list:
```

```
    match=re.findall(regex,i)
```

```
    if len(match)>0:
```

```
        print(i)
```

### OUTPUT:



```
PS G:\practicals> python -u "g:\practicals\dhyey_practicals.py"  
Rohil  
Shah  
snfajkdwakf  
Shamsheerj  
Raunak  
Rahil  
Nishith  
Harsh  
endwithj  
bdsjdbwj  
PS G:\practicals>
```