Python exp 1

Python id() Function

Python id() function returns an identity of an object. This is an integer which is guaranteed to be unique. This function takes an argument an object and returns a unique integer number which represents identity. Two objects with non-overlapping lifetimes may have the same id() value .

```python
class Student:
    def __init__(self, id, name):
        self.id = id
        self.name = name

student = Student(101,"Mohan")
print(student.id)
print(student.name)
# Calling function
val = id(student) # student class object
# Displaying result
print("Object id:",val)
```

Python type() Function

Python **type()** returns the type of the specified object if a single argument is passed to the type(). If three arguments are passed, it returns a new type of object.

```python
type(object, bases, dict)
List = [4, 5]
print(type(List))

Dict = {4: 'four', 5: 'five'}
print(type(Dict))
```

```

class Python:
    a = 0

InstanceOfPython = Python()
print(type(InstanceOfPython))
```

Python range() Function

Python **range()** function returns an immutable sequence of numbers starting from 0, increments by 1 and ends at a specified number.

Signature

```
range(start, stop, step)
```

Python exp 2

https://www.programiz.com/python-programming/operators

Python exp 3, 4, 5
https://www.programiz.com/python-programming/if-elif-else
https://www.programiz.com/python-programming/for-loop
https://www.programiz.com/python-programming/while-loop
https://www.programiz.com/python-programming/break-continue
https://www.programiz.com/python-programming/pass-statement
Python exp 6

```
size = 5
m = (2 * size) - 2
for i in range(0, size):
    A = 65
    for j in range(0, m):
        print(end=" ")
    m = m - 1
    for j in range(0, i + 1):
        print("%c " %(A), end="")
        A += 1
    print(" ")
```

Python exp 7

```
n = 5;
for i in range(n, 0, -1):
    a=0
    for j in range(i, n + 1, 1):
        print(abs(a+j-1),end = " ");
    print(" ");
for i in range(2, n + 1):
    for j in range(i, n + 1):
        print(a+j-1, end = ' ')
    print()
```

Python exp 8

```
n = 5;
m = (2 * n) - 2
for i in range(n, 0, -1):
    for j in range(0, m):
        print(end=" ")
    m = m - 1
    for j in range(i, n+1 , 1):
        print(chr(ord('A') + j - 1),end = " ")
    print(" ")
for i in range(2, n + 1):
    for j in range(-2,m):   # -2 coz spacing ka issues hai change maat karna pls
        print(end=" ")
    m = m + 1
    for j in range(i, n + 1):
        print(chr(ord('A') + j -1), end = ' ')
    print("  ")
```

Exp 9

```
"""f = open("T1.txt",'w')
a = input("Enter 10 words : ")
f.write(a)
f = open("T1.txt",'r')
str1=f.read()""" #if they asked to read froma  file
```

Exp 10

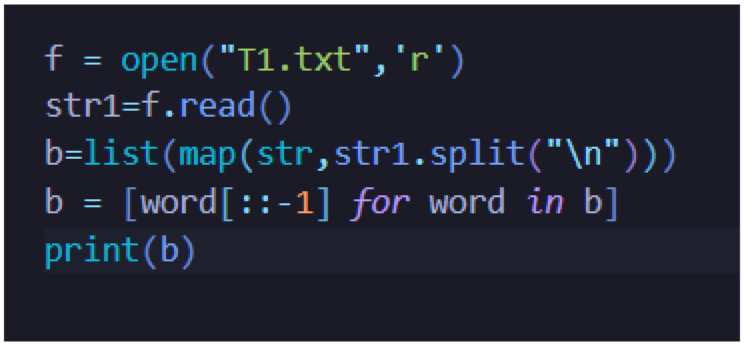
```
f = open("file.txt",'w')
```

```python
a = input("Enter 10 numbers : ")
f.write(a)
f = open("file.txt",'r')
str1=f.read()
b=list(map(int,str1.split(" ")))
# b.sort()
with open("t2.txt",'a+') as f1:
    sortedstr=" ".join(str(b))
    f1.write(sortedstr)
with open("t2.txt",'r') as f1:
    print(f1.read())
```

Exp 11

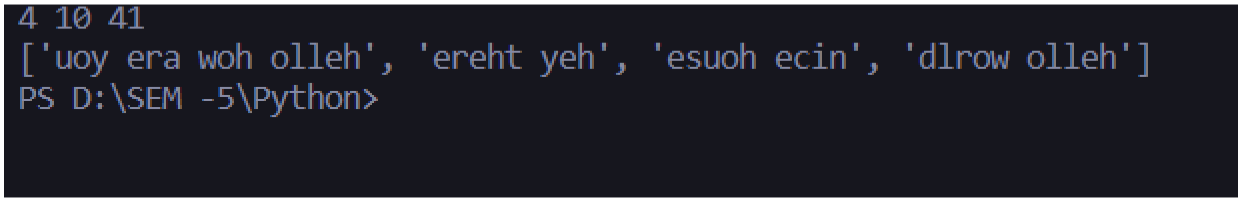
```python
f = open("T1.txt",'w')
a = input("Enter 10 words : ")
f.write(a)
f = open("T1.txt",'r')
str1=f.read()
# str1 = input("Enter 10 words : ")
d= dict()
for c in str1:
    if c in d:
        d[c] = d[c]+1
    else:
        d[c]=1
print(d)
```

Exp 12



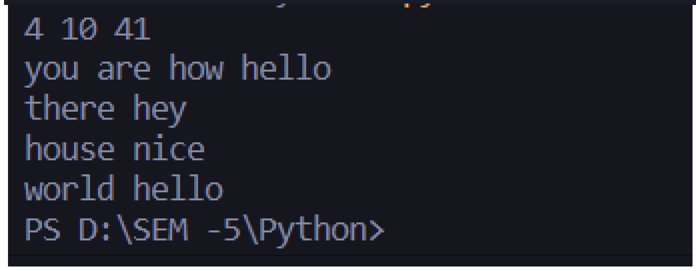
```python
f = open("T1.txt",'r')
str1=f.read()
b=list(map(str,str1.split("\n")))
b = [word[::-1] for word in b]
print(b)
```

Ans



```
4 10 41
['uoy era woh olleh', 'ereht yeh', 'esuoh ecin', 'dlrow olleh']
PS D:\SEM -5\Python>
```

```python
f = open("T1.txt",'r')
str1=f.read()
b=list(map(str,str1.split("\n")))
for word in b:
    words = word.split()
    words = list(reversed(words))
    print(" ".join(words))
```



```
4 10 41
you are how hello
there hey
house nice
world hello
PS D:\SEM -5\Python>
```

Exp 13

```python
f = open("test.txt",'r')
lines = 0
words = 0
characters = 0
for line in f:
    wordlist = line.split()
    lines += 1
    words += len(wordlist)
    for i in wordlist:
        for letter in i:
            if(letter.isspace):
                characters += 1
print( lines , words,characters)
```

Exp 14

fruits = ["apple" ,"orange","cherry"]
print(list(('apple','orange','cherry')))

```
print(len(fruits))
print(fruits.index("apple"))
fruits.append("grapes")
print(fruits)
fruits.insert(3,"banana")  # inserted at index 3
print(fruits)
print(fruits.count("apple"))
fruits.remove("grapes")
print(fruits)
print(fruits.pop())
fruits.reverse()
print(fruits)
fruits.sort()
print(fruits)
cpy = fruits.copy()
print("cpy -- " ,cpy)
print("clear cpy --",cpy.clear())
fruits.extend(list(('apple','orange','cherry')))
print(fruits)
```

Exp 15

```
t = (2,3,4,5,8,7,9,6,1,2,4)
t1 = (3,4,5,8,9,7,5,4,8,6,0)
print(len(t))
print(t.count(2))
print(t.index(8))
print(tuple(sorted(t)))
print(min(t))
print(max(t))
print cmp(t,t1)
print(tuple(reversed(t)))
```

Exp 16

NOTE:The discard() method removes the specified item from the set. This method is different from the remove() method, because **the remove() method will raise an error if the specified item does not exist, and the discard() method will not**

```
fruits = {"apple" ,"orange","cherry"}
```

```
more = {"guava","mango","cherry"}
fruits.add("banana")
print(fruits)
fruits.update(more)
print(fruits)
cpy = fruits.copy()
print("cpy --" , cpy)
print("clear --" , cpy.clear())
print(fruits.pop())
fruits.discard("grapes")
print(fruits)
fruits.remove("mango")
print(fruits)
print(fruits.union(more))
print(fruits.intersection(more))
print(fruits.difference(more))
```

EXP 17

```
dict1 = {1:"a",2:"b",3:"c",4:"d"}
dict2 = dict(name = "John", age = 36, country = "Norway")
print(len(dict1))
print(dict1.get(1))
print(dict2.pop("age"))
print(dict1.popitem())
print(dict1.keys())
print(dict1.values())
print(dict1.items())
dict2.update({"color": "White"})
print(dict2)
cpy = dict1.copy()
print(cpy)
print(cpy.clear())
```

EXP 18

```
# importing re library
import re
"""
1. The allowed characters are a-z, A-Z,0-9, #.
2. The first character should be a lower case alphabet symbol from a to k.
3. The second character should be a digit divisible by 3.
```

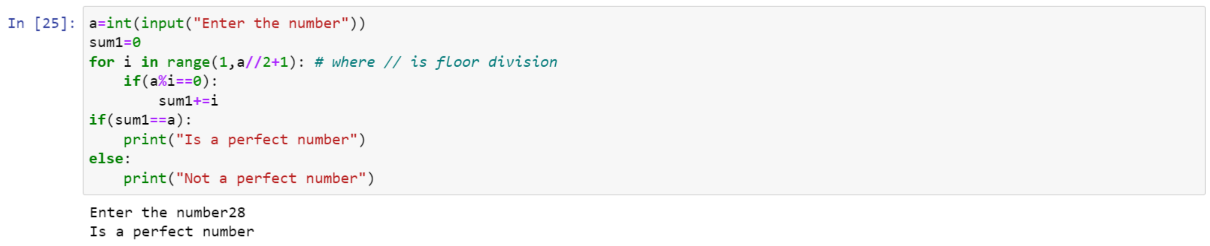
```
4. The length of identifier should be at least 2.
"""
def main():
    passwd = 'k3dxfcg'
    reg = "^[a-k][0369][a-zA-Z0-9#]*$"

    # compiling regex
    pat = re.compile(reg)

    # searching regex
    mat = re.search(pat, passwd)

    # validating conditions
    if mat:
        print("Password is valid.")
    else:
        print("Password invalid !!")

# Driver Code
if __name__ == '__main__':
    main()
```

Exp 19

```
import re # Importing re module
n=input('Enter Mobile number :')  # Reading input from the user
r=re.fullmatch('[7-9][0-9]{9}',n) # calling fullmatch function by passing pattern and n
if r!=None: # checking whether it is none or not
    print('Valid Number')
else:
    print('Not a valid number')
```

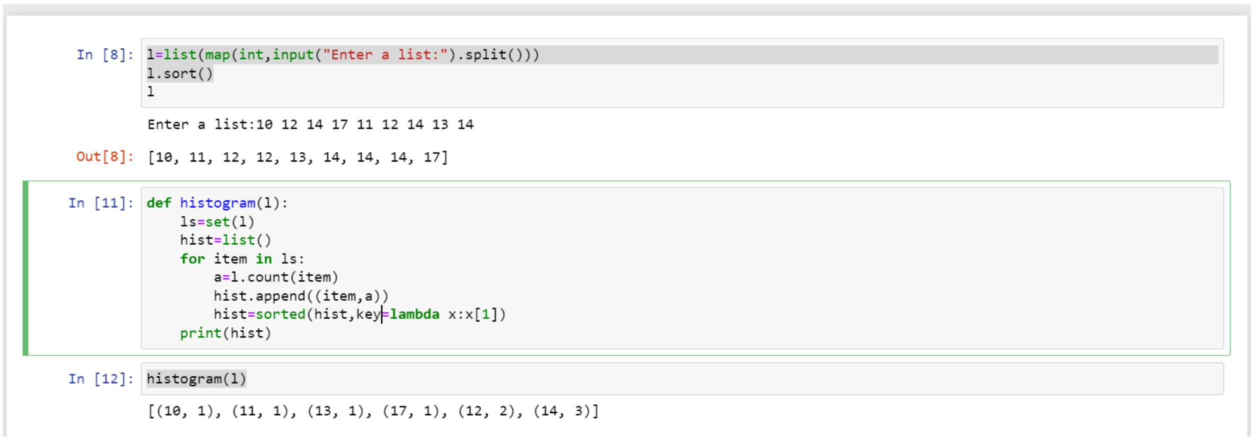
Exp 20

where // is floor division



```
In [25]: a=int(input("Enter the number"))
         sum1=0
         for i in range(1,a//2+1): # where // is floor division
             if(a%i==0):
                 sum1+=i
         if(sum1==a):
             print("Is a perfect number")
         else:
             print("Not a perfect number")

         Enter the number28
         Is a perfect number
```

Extra



```
In [8]: l=list(map(int,input("Enter a list:").split()))
        l.sort()
        l

        Enter a list:10 12 14 17 11 12 14 13 14

Out[8]: [10, 11, 12, 12, 13, 14, 14, 14, 17]

In [11]: def histogram(l):
             ls=set(l)
             hist=list()
             for item in ls:
                 a=l.count(item)
                 hist.append((item,a))
                 hist=sorted(hist,key=lambda x:x[1])
             print(hist)

In [12]: histogram(l)

         [(10, 1), (11, 1), (13, 1), (17, 1), (12, 2), (14, 3)]
```

Exp 21,22

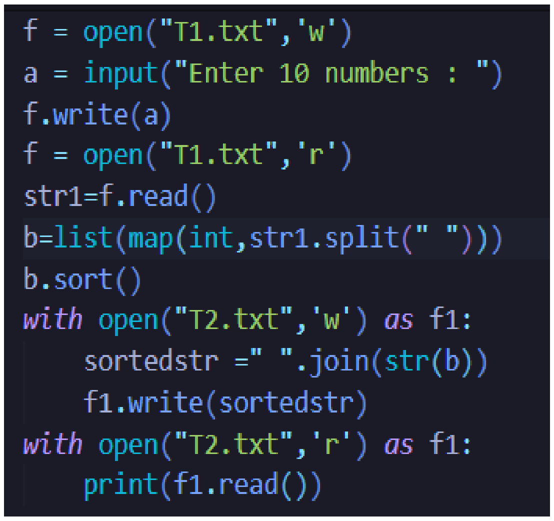
```
# tower of hanoi
n=int(input("Enter number of disks:"))

def towerofhanoi(n,start,middle,end):
    if n==1:
        print("Move the disk 1 from source:",start,"to destination:",end)
        print("end")
        return
    towerofhanoi(n-1,start,end,middle)
    print("Move disk",n," from source:",start,"to destination:",end)
    towerofhanoi(n-1,middle,start,end)

towerofhanoi(n,'A','B','C')


# lambda function to find greater of 2 inputs
a=int(input("a:"))
b=int(input("b:"))
ans=lambda a,b:a>b
print(ans(a,b))

# map function syntax:map(func, iter)
l1=[1,2,3,4]
l2=[2,4,6,8]
ans=map( lambda x,y:x+y,l1,l2)
print(list(ans))
```

```
## map and filter to find cube of all odd numbers in a list
l = [1,2,3,4,5,6,7,8,9,10]
odd_num = map(lambda x: x**3, filter(lambda   x: x%2!=0, l))
print(list(odd_num))
```

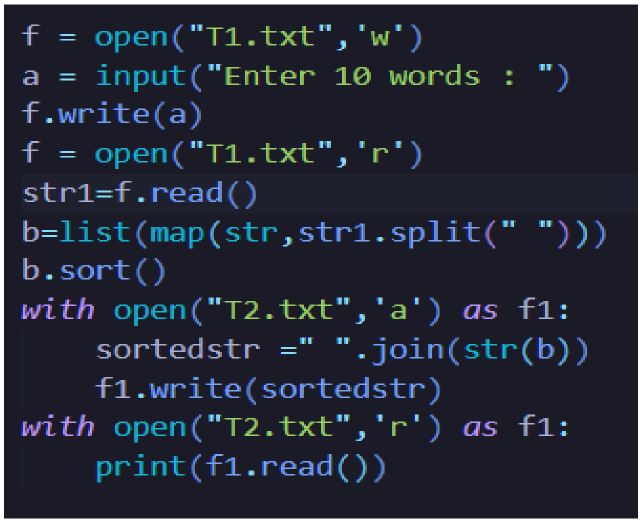
Exp 23

```
import uuid
class Employee:
    def __init__(self, name, idN,salary,designation):
        self.name = name
        self.idN = idN
        self.salary = salary
        self.designation = designation

class Developer(Employee):
    def __init__(self, name, idN,salary):
        super().__init__( name, idN,salary,"Developer")


class Tester(Employee):
    def __init__(self, name, idN,salary):
        super().__init__( name, idN,salary,"Tester")



class Manager(Employee):
    def __init__(self, name,salary):
        self.count = 1
        super().__init__( name, uuid.uuid1(),salary,"Manager")

    def addDeveloper(self,name,salary):
        self.count = self.count+1
        dev = Developer(name, uuid.uuid1(),salary)
        return dev

    def addTester(self,name,salary):
        self.count = self.count+1
        dev = Tester(name, uuid.uuid1(),salary)
        return dev
    def removeDeveloper(self,empDev):
        self.count = self.count-1
        del empDev

    def removeTester(self,empTest):
        self.count = self.count-1
        del empTest

man = Manager("Shubham",1000000000)
print(man.idN,man.designation,man.name)

dev= man.addDeveloper("Deevya",10000000)
print(dev.idN,dev.designation,dev.name)

test= man.addTester("Jenil",10000)
print(test.idN,test.designation,test.name)

total = man.count
print("Total :" , man.count)

removingDev = man.removeDeveloper(dev)

total = man.count
print("Total after removing dev:" , man.count)
```

Exp 24
See exp 4 of ur own files
Exp 25
ONLY DIFF IS ON LINE B=LIST(MAP(INT/STR …)
Even the line where with open("T2.txt", 'w' vs 'a') as f1:



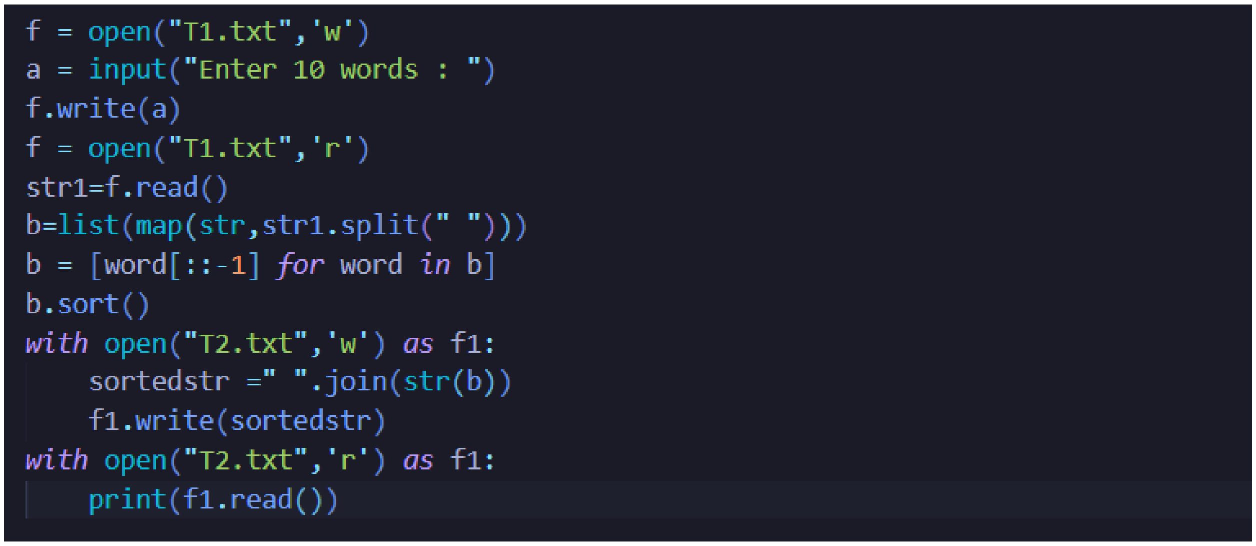
```python
f = open("T1.txt",'w')
a = input("Enter 10 numbers : ")
f.write(a)
f = open("T1.txt",'r')
str1=f.read()
b=list(map(int,str1.split(" ")))
b.sort()
with open("T2.txt",'w') as f1:
    sortedstr =" ".join(str(b))
    f1.write(sortedstr)
with open("T2.txt",'r') as f1:
    print(f1.read())
```



```python
f = open("T1.txt",'w')
a = input("Enter 10 words : ")
f.write(a)
f = open("T1.txt",'r')
str1=f.read()
b=list(map(str,str1.split(" ")))
b.sort()
with open("T2.txt",'a') as f1:
    sortedstr =" ".join(str(b))
    f1.write(sortedstr)
with open("T2.txt",'r') as f1:
    print(f1.read())
```

EXTRA

Take a T1.txt file with several words in it. Reverse each word and put the reversed words in order in a different file T2.txt.



```python
f = open("T1.txt",'w')
a = input("Enter 10 words : ")
f.write(a)
f = open("T1.txt",'r')
str1=f.read()
b=list(map(str,str1.split(" ")))
b = [word[::-1] for word in b]
b.sort()
with open("T2.txt",'w') as f1:
    sortedstr =" ".join(str(b))
    f1.write(sortedstr)
with open("T2.txt",'r') as f1:
    print(f1.read())
```

EXP 25

```python
import re
name,website,email,phone = [],[],[],[]
f = open(r"C:\Users\SHAH2H'\Desktop\folder\sem 5\python\test.txt",'r')
str = f.read()
email = re.findall(r'\S+@\S+', str)
```

```
website = re.findall(r'www.+in|www.+org|www.+com|plus.+com', str)
name = re.findall(r'Mr.+|Mrs.+', str)
phone = re.findall(r'\S[^a-zA-Z\n]+\d+\S[^a-zA-Z\n]', str)
print(f"\n Names are {name}\n Websites are {website}\n Email addresses are {email}\n
Phone addresses are {phone}\n")
```

EXP 26

```
import mysql.connector
import mysql.connector
from mysql.connector import Error
constring = mysql.connector.connect(
    host='localhost',
    database='job',
    user='root',
    password='suchi#1579',
)

cursor = constring.cursor()
TABLES = {}
while(1):
    print("\nWelcome to database\n\nThe tables are: ")
    cursor.execute("Show tables")
    for table in cursor:
        print(table[0])
    print("\n1)Create Table\n2)Insert into table\n3)Delete a row\n4)Display all
rows\n5)Update a row\n6)Search a record\n7)Exit")
    val = int(input("Your choice: "))
    print()
    if val == 1:
        string = input("Enter name of the table: ")
        print("It has 3 columns id, title, description")
        sql = f"Create table {string}(`id` int(5) not null auto_increment,`title` varchar(50),
`description` varchar(50), PRIMARY KEY (`id`));"
```

```
    cursor.execute(sql)
    print("Table created successfully!")
elif val == 2:
    string = input("Enter name of the table: ")
    tile = input("Enter the title: ")
    desc = input("Enter the description: ")
    val = (tile,desc)
    sql = f"Insert into {string} (`title`,`description`) values (%s,%s);"
    cursor.execute(sql,val)
    constring.commit()
    print("Values added successfully!")
elif val ==3:
    string = input("Enter name of the table: ")
    tile = input("Enter the title: ")
    desc = input("Enter the description: ")
    val = (tile,desc)
    sql = f"delete from {string} where `title` = %s and `description` = %s;"
    cursor.execute(sql,val)
    constring.commit()
    print("Values removed successfully!")
elif val == 4:
    string = input("Enter name of the table: ")
    sql = f"Select * from {string}"
    cursor.execute(sql)
    for row in cursor:
        print(row)
elif val == 5:
    string = input("Enter name of the table: ")
    id = int(input("Enter ID of row to be changed: "))
    tile = input("Enter new title: ")
    desc = input("Enter new description: ")
    val = (tile,desc,id)
    sql = f"update {string} set `title` = %s , `description` = %s where `id` = %s;"
```

```
        cursor.execute(sql,val)
        constring.commit()
        print("Values updated successfully!")
    elif val == 6:
        string = input("Enter name of the table: ")
        tile = input("Enter title to be searched: ")
        val = (tile,)
        sql = f"Select * from {string} where `title` = %s;"
        cursor.execute(sql,val)
        print("Search Result: ")
        for row in cursor:
            print(row)
    else:
        cursor.close()
        break
    # for table_name in cursor:
    #     print(table_name)
```

EXP 27 DONE

EXP 28

CALCULATOR KI SPELLING PROPER LIKHNA

```
import tkinter as tk
import tkinter.messagebox
from tkinter.constants import SUNKEN

window=tk.Tk()
window.title('Calculator')
frame=tk.Frame(master=window,bg="skyblue",padx=10)
frame.pack()
entry=tk.Entry(master=frame,relief=SUNKEN,borderwidth=3,width=30)
entry.grid(row=0,column=0,columnspan=3,ipady=2,pady=2)
```

```
def myclick(number):
        entry.insert(tk.END,number)

def equal():
        try:
                y=str(eval(entry.get()))
                entry.delete(0,tk.END)
                entry.insert(0,y)
        except:
                tkinter.messagebox.showinfo("Error","Syntax Error")

def clear():
        entry.delete(0,tk.END)

button_1=tk.Button(master=frame,text='1',padx=15,pady=5,width=3,command=lambda:
myclick(1))
button_1.grid(row=1,column=0,pady=2)
button_2=tk.Button(master=frame,text='2',padx=15,pady=5,width=3,command=lambda:
myclick(2))
button_2.grid(row=1,column=1,pady=2)
button_3=tk.Button(master=frame,text='3',padx=15,pady=5,width=3,command=lambda:
myclick(3))
button_3.grid(row=1,column=2,pady=2)
button_4=tk.Button(master=frame,text='4',padx=15,pady=5,width=3,command=lambda:
myclick(4))
button_4.grid(row=2,column=0,pady=2)
button_5=tk.Button(master=frame,text='5',padx=15,pady=5,width=3,command=lambda:
myclick(5))
button_5.grid(row=2,column=1,pady=2)
button_6=tk.Button(master=frame,text='6',padx=15,pady=5,width=3,command=lambda:
myclick(6))
button_6.grid(row=2,column=2,pady=2)
```

```
button_7=tk.Button(master=frame,text='7',padx=15,pady=5,width=3,command=lambda:
myclick(7))
button_7.grid(row=3,column=0,pady=2)
button_8=tk.Button(master=frame,text='8',padx=15,pady=5,width=3,command=lambda:
myclick(8))
button_8.grid(row=3,column=1,pady=2)
button_9=tk.Button(master=frame,text='9',padx=15,pady=5,width=3,command=lambda:
myclick(9))
button_9.grid(row=3,column=2,pady=2)
button_0=tk.Button(master=frame,text='0',padx=15,pady=5,width=3,command=lambda:
myclick(0))
button_0.grid(row=4,column=1,pady=2)

button_add=tk.Button(master=frame,text="+",padx=15,pady=5,width=3,command=lamb
da:myclick('+'))
button_add.grid(row=5,column=0,pady=2)

button_subtract=tk.Button(master=frame,text="-",padx=15,pady=5,width=3,command=la
mbda:myclick('-'))
button_subtract.grid(row=5,column=1,pady=2)

button_multiply=tk.Button(master=frame,text="*",padx=15,pady=5,width=3,command=la
mbda:myclick('*'))
button_multiply.grid(row=5,column=2,pady=2)

button_div=tk.Button(master=frame,text="/",padx=15,pady=5,width=3,command=lambd
a:myclick('/'))
button_div.grid(row=6,column=0,pady=2)

button_clear=tk.Button(master=frame,text="clear",padx=15,pady=5,width=12,command=
clear)
button_clear.grid(row=6,column=1,columnspan=2,pady=2)
```

```
button_equal=tk.Button(master=frame,text="=",padx=15,pady=5,width=9,command=equ
al)
button_equal.grid(row=7,column=0,columnspan=3,pady=2)

window.mainloop()
```

EXP 29 NO NEED TO STUDY UK

EXP 30 NO NEED TO STUDY UK