Experiment 5

Frason Francis / 201903020 / SE-IT

(A) Designing Graphical user interface (GUI) using built-in tools PyQt GUI database connectivity to perform CRUD operations in python.

database.py

```
import tkinter as tk
from tkinter import Button

import database
from database import *

mainWindow = tk.Tk()
mainWindow.title("add data")

mainheadinglabel=tk.Label(mainWindow,text='STUDENT MAGANGEMENT
SYSTEM',padx=(30),pady=(30), fg='red')
mainheadinglabel.grid(row=0 , column=2)

headinglabel1=tk.Label(mainWindow,text='name',padx=(10),pady=(10))
headinglabel1.grid(row=1 , column=0)

name_field=tk.Entry(mainWindow)
name_field.grid(row=1 , column=3,padx=(10),pady=(10))


headinglabel2=tk.Label(mainWindow,text='college name',padx=(10),pady=(10))
headinglabel2.grid(row=2 , column=0)

college_field=tk.Entry(mainWindow )
college_field.grid(row=2 , column=3,padx=(10),pady=(10))

headinglabel3=tk.Label(mainWindow,text='address',padx=(10),pady=(10))
headinglabel3.grid(row=3 , column=0)

address_field=tk.Entry(mainWindow)
address_field.grid(row=3 , column=3,padx=(10),pady=(10))

headinglabel4=tk.Label(mainWindow,text='phone',padx=(10),pady=(10))
headinglabel4.grid(row=4 , column=0)

phone_field=tk.Entry(mainWindow)
phone_field.grid(row=4 , column=3,padx=(10),pady=(10))
def get():
    name1=name_field.get()
    college1=college_field.get()
    address1=address_field.get()
```

```
  phone1=phone_field.get()
  insert(name1,college1,address1,phone1)
  name_field.delete(0,tk.END)
  college_field.delete(0,tk.END)
  address_field.delete(0,tk.END)
  phone_field.delete(0,tk.END)




insert1=tk.Button(mainWindow,text='insert',command =lambda : get())
insert1.grid(row=5 , column=0)

show1=tk.Button(mainWindow,text='show',command =lambda : show())
show1.grid(row=5,column=3)

mainWindow.mainloop()
```

fronthand.py

```
import tkinter as tk
from tkinter import Button

import database
from database import *

mainWindow = tk.Tk()
mainWindow.title("add data")

mainheadinglabel=tk.Label(mainWindow,text='STUDENT MAGANGEMENT
SYSTEM',padx=(30),pady=(30), fg='red')
mainheadinglabel.grid(row=0 , column=2)

headinglabel1=tk.Label(mainWindow,text='name',padx=(10),pady=(10))
headinglabel1.grid(row=1 , column=0)

name_field=tk.Entry(mainWindow)
name_field.grid(row=1 , column=3,padx=(10),pady=(10))


headinglabel2=tk.Label(mainWindow,text='college name',padx=(10),pady=(10))
headinglabel2.grid(row=2 , column=0)

college_field=tk.Entry(mainWindow )
college_field.grid(row=2 , column=3,padx=(10),pady=(10))

headinglabel3=tk.Label(mainWindow,text='address',padx=(10),pady=(10))
headinglabel3.grid(row=3 , column=0)

address_field=tk.Entry(mainWindow)
address_field.grid(row=3 , column=3,padx=(10),pady=(10))
```

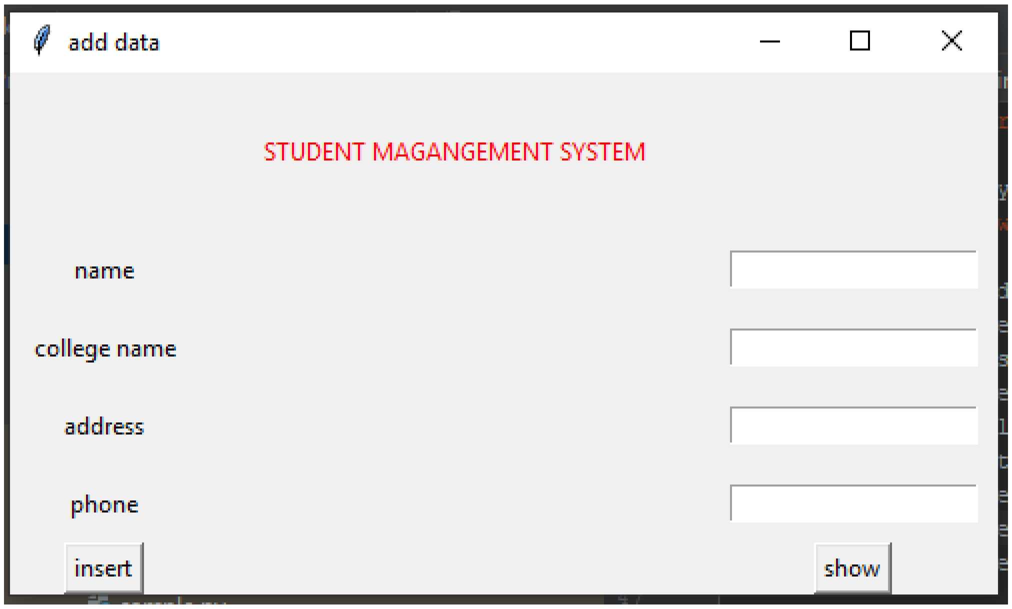
```
headinglabel4=tk.Label(mainWindow,text='phone',padx=(10),pady=(10))
headinglabel4.grid(row=4 , column=0)

phone_field=tk.Entry(mainWindow)
phone_field.grid(row=4 , column=3,padx=(10),pady=(10))
def get():
   name1=name_field.get()
   college1=college_field.get()
   address1=address_field.get()
   phone1=phone_field.get()
   insert(name1,college1,address1,phone1)
   name_field.delete(0,tk.END)
   college_field.delete(0,tk.END)
   address_field.delete(0,tk.END)
   phone_field.delete(0,tk.END)

insert1=tk.Button(mainWindow,text='insert',command =lambda : get())
insert1.grid(row=5 , column=0)

show1=tk.Button(mainWindow,text='show',command =lambda : show())
show1.grid(row=5,column=3)

mainWindow.mainloop()
```



(B) Different File Handling operations in Python.

Files

Python uses file objects to interact with the external files on your computer. These file objects can be of any file format on your computer i.e. can be an audio file, a text file, emails, Excel documents, etc.

Python has a built-in open function that allows us to open and play with basic file types.

iPython Writing a File

In [1]:

```
%%writefile test.txt
Hello, this is a quick test file hjgtyudfyffhgghghhfch
```

```
Writing test.txt
```

In [2]:

```
pwd()
```

Out[2]:

```
'C:\\Users\\jkfra\\Downloads'
```

Python Opening a file

We can open a file with the open() function. This function also takes in arguments (also called parameters). Let's see how this is used:

In [3]:

```
# Open the text.txt we made earlier
my_file = open('test.txt')
```

In [66]:

```
# We can now read the file
my_file.read()
```

Out[66]:

```
'Hello, this is a quick test file hjgtyudfyffhgghghhfch\n'
```

In [64]:

```
# But what happens if we try to read it again?
my_file.read()
```

Out[64]:

```
''
```

This happens because you can imagine the reading "cursor" is at the end of the file after having read it. So there is nothing left to read. We can reset the "cursor" like this:

In [72]:

```
# Seek to the start of file (index 0)
my_file.seek(20)
```

Out[72]:

```
20
```

In [73]:

```
# Now read again
my_file.read()
```

Out[73]:

```
'ck test file hjgtyudfyffhgghghhfch\n'
```

In order to not have to reset every time, we can also use the readlines method. Use caution with large files, since everything will be held in memory. We will learn how to iterate over large files later in the course.

In [40]:

```
# Seek to the start of file (index 0)
my_file.seek(0)
```

Out[40]:

```
0
```

In [41]:

```
# Readlines returns a list of the lines in the file.
my_file.readlines()
```

Out[41]:

```
['Hello, this is a quick test file']
```

Writing to a File

By default, using the open() function will only allow us to read the file, we need to pass the argument 'w' to write over the file. For example:

In [74]:

```
# Add the second argument to the function, 'w' which stands for write
my_file = open('test.txt','w+')
```

In [75]:

```
# Write to the file
my_file.write('This is a new line')
```

Out[75]:

```
18
```

In [76]:

```
# Seek to the start of file (index 0)
my_file.seek(0)
```

Out[76]:

```
0
```

In [77]:

```
# Read the file
my_file.read()
```

Out[77]:

```
'This is a new line'
```

Iterating through a File

Let's get a quick preview of a for loop by iterating over a text file. First, let's make a new text file with some iPython Magic:

In [78]:

```
%%writefile test.txt
First Line
Second Line
```

```
Overwriting test.txt
```

In [79]:

```
my_file = open('test.txt')
my_file.read()
```

Out[79]:

```
'First Line\nSecond Line\n'
```

Now we can use a little bit of flow to tell the program to for through every line of the file and do something:

In [80]:

```
for line in open('test.txt'):
    print(line)
```

```
First Line

Second Line
```

In [50]:

```
# Pertaining to the first point above
for asdf in open('test.txt'):
    print(asdf)
```

```
First Line

Second Line
```

StringIO

The StringIO module implements an in-memory filelike object. This object can then be used as input or output to most functions that would expect a standard file object.

The best way to show this is by example:

In [83]:

```
from io import StringIO
```

In [84]:

```
# Arbitrary String
message = 'This is just a normal string.'
```

In [85]:

```
# Use StringIO method to set as file object
f = StringIO(message)
```

Now we have an object *f* that we will be able to treat just like a file. For example:

In [86]:

```
f.read()
```

Out[86]:

```
'This is just a normal string.'
```

We can also write to it

In [87]:

```
f.write(' Second line written to file like object')
```

Out[87]:

```
40
```

In [88]:

```
# Reset cursor just like you would a file
f.seek(5)
```

Out[88]:

```
5
```

In [89]:

```
# Read again
f.read()
```

Out[89]:

```
'is just a normal string. Second line written to file like object'
```

In [ ]:

```

```