INT-213 (PYTHON PROGRAMMING)

REPORT FOR QUIZ MODULE

NAME	: ADITYA RAJ
REGISTRATION NUMBER	: 12008354
NAME	: SRINAND KAKKATTIL
REGITRATION NUMBER	: 12008040
PROGRAM	: B.TECH CSE
SCHOOL	: SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY	:LOVELY PROFESSIONAL UNIVERSITY
DATE OF SUBMISSION	:20th NOVEMBER 2021

TEAM MEMBERS: -

TEAM LEADER:

ADITYA RAJ

Contributions:

1.Coding(joined)

2.GUI (joined)

3.Report of Project(joined)

SRINAND KAKKATTIL

Contributions:

1.Coding(joined)

2.GUI(joined)

3.Report of project(joined)

ABSTRACT: -

This project has been done as part of my course for the CSE at Lovely Professional University, supervised by Ankita Wadhawan. We had two months to complete this project.

The project topic is assigned to us was to make a quiz module with a proper interface which has categories easy, average and tough.

In this project we take test in proper interface, result will show at end.

First, we take details of students which will saved in our system. Then they enter in quiz module and give test. At last, we show their result. In this project we also get to know how much scored in the test.

ACKNOWLEDGEMENT: -

I would like to express special thanks of gratitude to my teacher as well as our university who gave me the golden opportunity to do this wonderful project on the topic Quiz Module which also help me doing a lot of research and I came to know about so many new things. I am thankful to them. Secondly, I would like to thank my parents and friends who helped me a lot of finalizing this project within the limited time frame.

I sincerely thank everyone for the guidance and encouragement in carrying out this project work I also wish to express my gratitude to the official and other staff members who rendered their help during the period of my project work.

INTRODUCTION

PYTHON

Python language introduction

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Python features

Python's features include –

• Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

• Easy-to-read − Python code is more clearly defined and visible to the eyes. • Easy-to-maintain − Python's source code is fairly easy-tomaintain.

• A broad standard library − Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh.

• Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

• Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

• Extendable − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

• Databases − Python provides interfaces to all major commercial databases.

• GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Python graphical user interfaces (GUIs)

- **Tkinter** – Tkinter is the Python interface to the Tk GUI toolkit shipped with Python.

- **wxPython** – This is an open-source Python interface for wxWindows.

- **JPython** – JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine.

There are many other interfaces also.

GUI used

Tkinter GUI used in python for making scientific calculator.

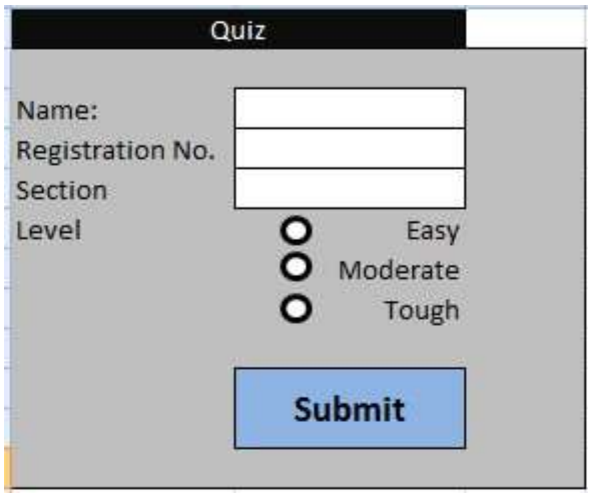
Tkinter Widgets

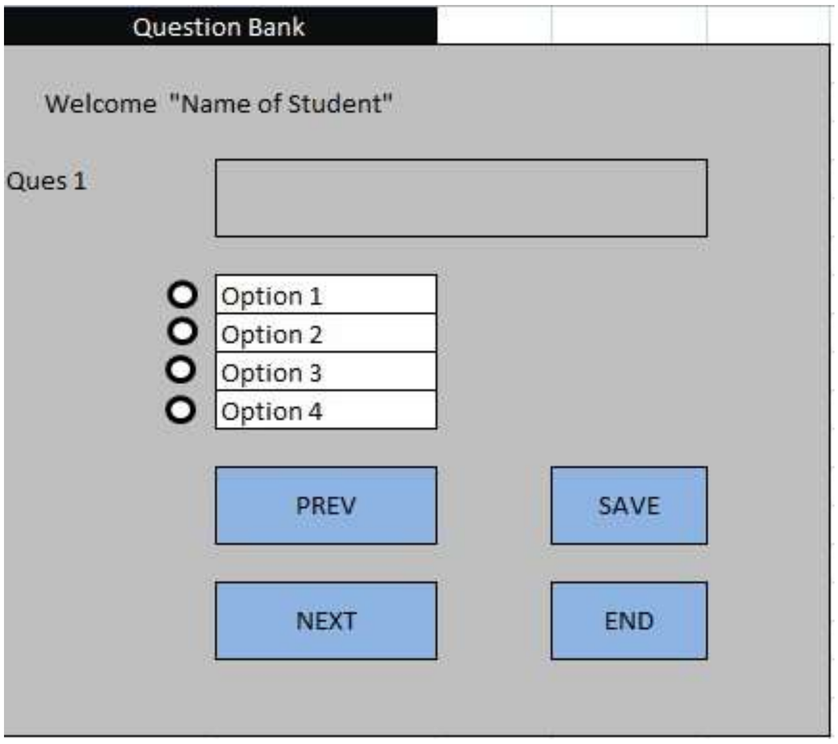
Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets. There are some types of widgets in Tkinter. We present these widgets as well as a brief description in the following table–

Operator & Description
<u>Button</u> The Button widget is used to display buttons in your application.
<u>Entry</u> The Entry widget is used to display a single-line text field for accepting values from a user.
<u>Frame</u> The Frame widget is used as a container widget to organize other widgets.
<u>Label</u> The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
<u>Menubutton</u> The Menubutton widget is used to display menus in your application.
<u>Menu</u> The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.

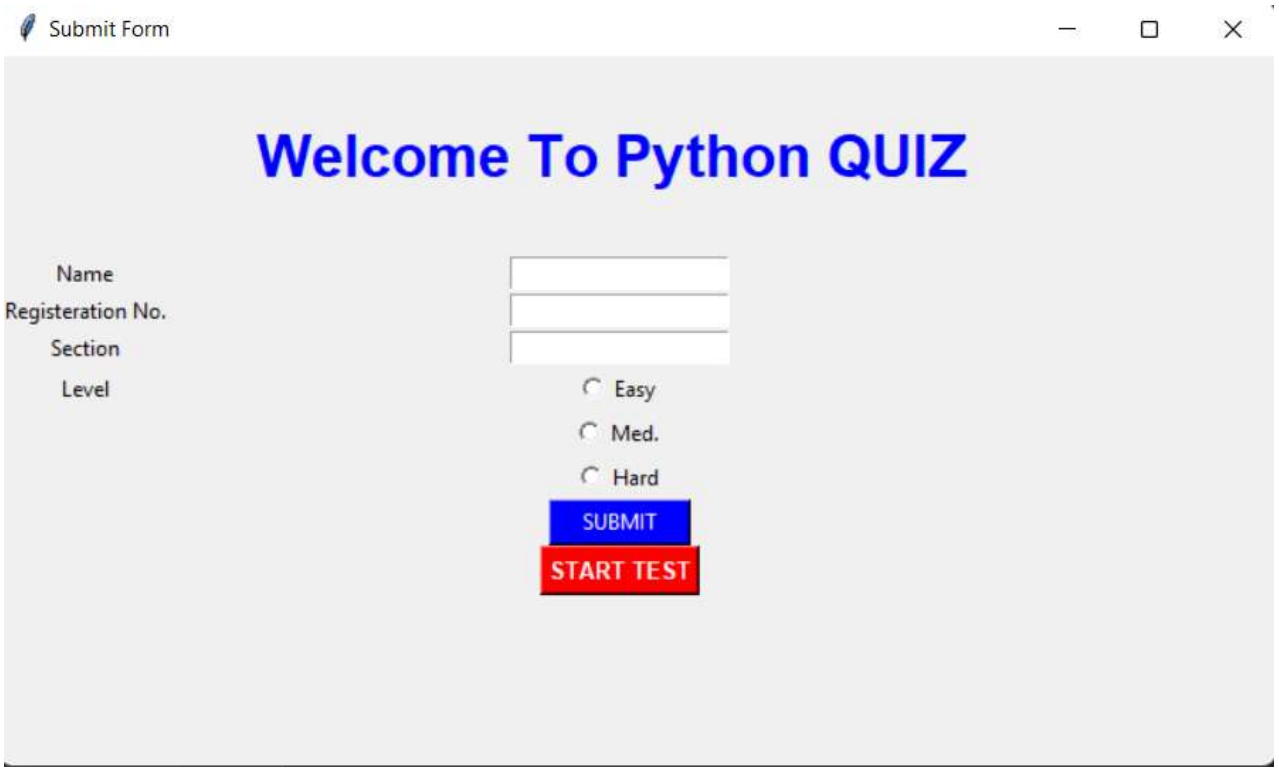
Message The Message widget is used to display multiline text fields for accepting values from a user.
Text The Text widget is used to display text in multiple lines.
LabelFrame A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
tkMessageBox This module is used to display message boxes in your applications.

SCREENSHOTS





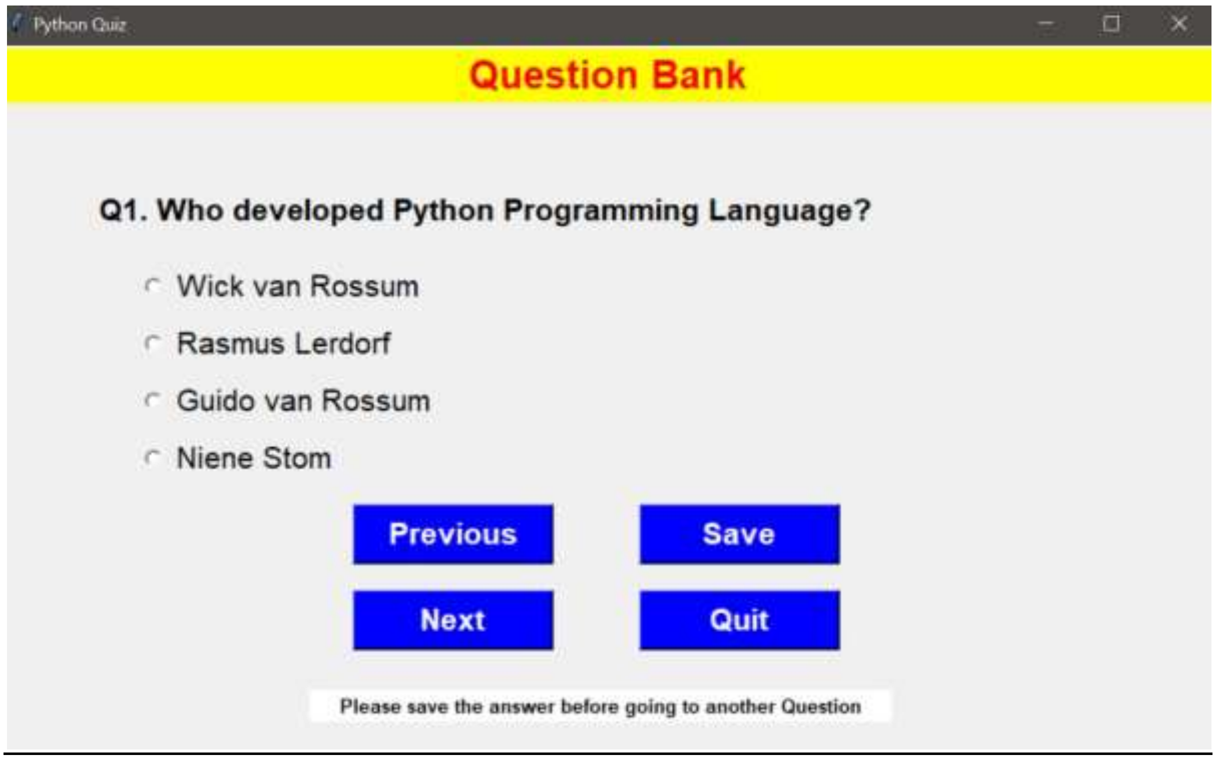
SCREENSHOTS OF THE OUTPUT:


Submit Form

Welcome To Python QUIZ

Name	
Registeration No.	
Section	
Level	Easy
	Med.
	Hard

SECOND PAGE:


Python Quiz

Question Bank

Q1. Who developed Python Programming Language?

- Wick van Rossum
- Rasmus Lerdorf
- Guido van Rossum
- Niene Stom

Please save the answer before going to another Question

Code: -

Part 1

```
import os
from tkinter import *

gui = Tk()
var=IntVar()
def getvals():
    print("Submitting form")

    print(f"{namevalue.get(), registervalue.get(), sectionvalue.get()} ")

    with open("records.txt", "a") as f:
        f.write(f"{namevalue.get(), registervalue.get(), sectionvalue.get()}\n ")

gui.geometry("720x400")
#Heading
Label(gui, text="Welcome To Python QUIZ ", font="comicsansms 25 bold", pady=35, width=25, fg="blue").grid(row=0, column=3)

#Text for our form
name = Label(gui, text="Name")
register = Label(gui, text="Registeration No.")
section = Label(gui, text="Section")
level=Label(gui,text="Level")

#Pack text for our form
name.grid(row=1, column=2)
register.grid(row=2, column=2)
section.grid(row=3, column=2)
level.grid(row=4,column=2)

# Tkinter variable for storing entries
namevalue = StringVar()
registervalue = StringVar()
sectionvalue = StringVar()


#Entries for our form
nameentry = Entry(gui, textvariable=namevalue)
registerentry = Entry(gui, textvariable=registervalue)
sectionentry = Entry(gui, textvariable=sectionvalue)

# Packing the Entries
nameentry.grid(row=1, column=3)
registerentry.grid(row=2, column=3)
sectionentry.grid(row=3, column=3)
Radiobutton(gui,text="Easy",padx= 20, variable= var, value=1).grid(row=4,column=3)
Radiobutton(gui,text="Med.",padx= 20, variable= var, value=2).grid(row=5,column=3)
Radiobutton(gui,text="Hard",padx= 20, variable= var, value=3).grid(row=6,column=3)
Button(text="SUBMIT", command=getvals,width=10,bg="blue", fg="white").grid(row=7, column=3)

#To start the Test
def run():
        os.system('main.py')
#Button & packing it and assigning it a command
Button(text="START TEST", command=run,width=10, bg="red", fg="white", font=("ariel", 10, " bold")).grid(row=9, column=3)

# set the title of the Window
gui.title("Submit Form")

# Start the GUI
gui.mainloop()
# END OF THE PROGRAM
```

Part 2

```python
from tkinter import *
from tkinter import messagebox as mb

import json
# class to define the components of the GUI
class Quiz:

    def __init__(self):
        # set question number to 0
        self.q_no = 0
        self.q_nos = -1
        # assigns ques to the display_question function to update later.
        self.display_title()
        self.display_notice()
        self.display_question()

        # opt_selected holds an integer value which is used for
        # selected option in a question.
        self.opt_selected = IntVar()

        # displaying radio button for the current question and used to
        # display options for the current question
        self.opts = self.radio_buttons()

        # display options for the current question
        self.display_options()

        # displays the button for next and exit.
        self.buttons()

        # no of questions
        self.data_size = len(question)

        # keep a counter of correct answers
        self.correct = 0
    def display_result(self):
        # calculates the wrong count
        wrong_count = self.data_size - self.correct
        correct = f"Correct: {self.correct}"
        wrong = f"Wrong: {wrong_count}"

        # calcultaes the percentage of correct answers
        score = int(self.correct / self.data_size * 100)
        result = f"Score: {score}%"

        # Shows a message box to display the result
        mb.showinfo("Result", f"{result}\n{correct}\n{wrong}")

    # This method checks the Answer after we click on Next.
    def check_ans(self, q_no):

        # checks for if the selected option is correct
        if self.opt_selected.get() == answer[q_no]:
            # if the option is correct it return true
            return True

    def next_btn(self):
        # Moves to next Question by incrementing the q_no counter
        self.q_no += 1

        # checks if the q_no size is equal to the data size
        if self.q_no == self.data_size:
            # if it is correct then it displays the score
            self.display_result()

            # destroys the GUI
            gui.destroy()
        else:
            # shows the next question
            self.display_question()
            self.display_options()


    def previous_btn(self):
        # Moves to previous Question by decrementing the q_no counter
        self.q_no -= 1

        # checks if the q_no size is equal to first question
        if self.q_no == self.q_nos:
            # destroys the GUI
            gui.destroy()
        else:
            # shows the next question
            self.display_question()
            self.display_options()

    def save_btn(self):
        # Check if the answer is correct
        if self.check_ans(self.q_no):
            # if the answer is correct it increments the correct by 1
            self.correct += 1

    def buttons(self):
        next_button = Button(gui, text="Next", command=self.next_btn,
                             width=10, bg="blue", fg="white", font=("ariel", 16, "bold"))
        next_button.place(x=250, y=380)

        previous_button = Button(gui, text="Previous", command=self.previous_btn,
                             width=10, bg="blue", fg="white", font=("ariel", 16, "bold"))

        previous_button.place(x=250, y=320)

        save_button = Button(gui, text="Save", command=self.save_btn,
                             width=10, bg="blue", fg="white", font=("ariel", 16, "bold"))
        save_button.place(x=450, y=320)

        quit_button = Button(gui, text="Quit", command=gui.destroy,
                             width=10, bg="blue", fg="white", font=("ariel", 16, " bold"))
        quit_button.place(x=450, y=380)


    def display_options(self):
        val = 0
        self.opt_selected.set(0)
        # text of the radio buttons.
        for option in options[self.q_no]:
            self.opts[val]['text'] = option
            val += 1

    # This method shows the current Question on the screen
    def display_question(self):
```

```
    # This method shows the current Question on the screen
    def display_question(self):

        # setting the Question properties
        q_no = Label(gui, text=question[self.q_no], width=60,
                     font=('ariel', 16, 'bold'), anchor='w')

        # placing the option on the screen
        q_no.place(x=70, y=100)

    # This method is used to Display Title
    def display_title(self):

        # The title to be shown
        title = Label(gui, text="Question Bank",
                      width=50, bg="yellow", fg="red", font=("ariel", 20, "bold"))

        # place of the title
        title.place(x=0, y=2)


    def display_notice(self):

        # The notice to be shown at the end
        title = Label(gui, text="Please save the answer before going to another Question",
                      width=50, bg="white", fg="black", font=("ariel", 10, "bold"))

        # place of the notice
        title.place(x=220, y=450)

    def radio_buttons(self):
        q_list = []
        y_pos = 150
        # adding the options to the list
        while len(q_list) < 4:
            # setting the radio button properties
            radio_btn = Radiobutton(gui, text=" ", variable=self.opt_selected,
                                    value=len(q_list) + 1, font=("ariel", 16))

            # adding the button to the list
            q_list.append(radio_btn)

            # placing the button
            radio_btn.place(x=100, y=y_pos)

            # incrementing the y-axis position by 40
            y_pos += 40

        # return the radio buttons
        return q_list

# Create a GUI Window
gui = Tk()

# set the size of the GUI Window
gui.geometry("850x500")

# set the title of the Window
gui.title("Python Quiz")

# get the data from the json file
with open('data.json.txt') as f:
    data = json.load(f)

# set the question, options, and answer
question = (data['question'])
options = (data['options'])
answer = (data['answer'])

# create an object of the Quiz Class.
quiz = Quiz()

# Start the GUI
gui.mainloop()

# END OF THE PROGRAM
```

CONCLUSION: -

This project has really been faithful and informative. It has made us learn and understand the many trivial concepts of Python Language. As we have used python Tkinter as a GUI it provides various controls, such as buttons, labels, and text boxes to build a user-friendly application. The fast-growing use of internet confirms the good future and scope of the proposed project. Finally, it has taught us a valuable lifelong lesson about the improvements and working and interacting in a group.

While doing this project, we learn lot of things like GUI, python programming, logics.
We understand teamwork and while doing this project we enjoyed a lot.

REFERENCE: -

1. Python Tutorial (w3schools.com)
2. Untitled - Jupyter Notebook (turing.ac.uk)
3. Stack Overflow - Where Developers Learn, Share, & Build Careers
4. Python Tutorial (tutorialspoint.com)