# Python code course

## Project Oriented Programming

# Stacey Loulouse

# Learn How to code

## [HERE](https://www.youtube.com/watch?v=PctPEo8-QkY)

# What You Will Learn

## **Project oriented programming**

## Factors of a coding project

There are five things to consider when working on a project. It is important to realize that every good project solves a problem. The first step is to recognize what that problem entails. The problem statement shows that you know what the over-arching response to a need is. Some of the greatest developers spend time crunching code to find that they get off track when the problem statement is not clearly defined. The next step is to define the scope. Scope can be looked at as the borders of the program. The end-result should have x-y-z qualities. If not, then there is still more development to do. The developer should not be so much concerned with how to code it at first. The how-to code does not actually appear until you understand what tools and technologies are available or at your disposal. When working with a company, these are usually chosen for you. We will utilize a limited number of tools and technologies in this text so that you become familiar with Python and the modules and packages that help to develop the code that will bring us to a resulting application.

## Answer A Problem

## Define the Scope

## Know and Select Tools

## Collaborate with Others

## create & implement deployment plan

## NOT starting with “Hello World”

All too often textbooks start with the basics by showing the minute details of the language. This book will fast-track the developer into a mode of learning that encompasses the basics of engineering. Rather than starting off with a simple statement such as print (‘hello world’), we will look at the way development should progress first.

## Game Development

The second project will cover some elements of game development such as logic, game progression, and scoring management.

## Database Management

We will briefly cover SQLITE3 database management in the final capstone project. After creating a database and table of information, we will be able to perform basic functions on the data to be stored. The basic database functions will be to insert, update, and delete information.

## Web Development

This book will not go into much detail of web development except to connect website information into a python project such as the fourth capstone project. You will see more about this in the last part of this book.

# When You Will Learn

## **At Your Own Pace**

## Live zoom sessions

# meeting id: 317-920-0956

# passcode: 058021

## Tuesdays @ 2pm

## Fridays @ 5 pm

## Sundays @ 7 pm

# How You Will Learn

## **At Your Own Pace**

## 1 to 3 live Sessions per week

## Instructor-Led

## Textbook guide

## **Code Review**

## Live Q&A

## **Hands-On Exercises**

## Project Assessments

## Google

## **Documentation Research**

# Project Resource List

Before we begin to code a single line of instruction, we need to set the environment for coding. Then we use the resources listed below to accomplish the tasks or requirements of the project. GitHub is a version control and collaboration tool that allows you to share code, keep track of changes and host code storage called a repository. When you save a copy of this online storage system, it is called a clone. Each clone has a website identifier or URL. We will look more closely at the process of getting this environment set up so that you can access the code and update GitHub repositories from your VS code editor. Do not worry if you have never heard of these terms before. It will become clear with the links and images provided in this text. If you can follow directions, then you will be successful in this course.

Aside from Python, you will be downloading GitHub, GitBash and VS Code editor. The other resources mentioned are modules within Python that will be used throughout the course. It is a good idea to become familiar with the names of these modules and packages.

## GitHub

## Gitbash

## python

## VS Code editor

## Tkinter

## random

## playsound

## Pillow (PIL), ImageTk, Image

## Django

## HTML

## SQLite3

# **Project Line Up**

## First Project – Basic calculator

## Second Project – Prime Time Game

## Third Project – Photo Gallery

## Fourth Project – Widgets (Beginner Level Capstone)



# For BeginnersComing up in the next book…

## Fifth project – email sender

## Sixth Project – Game of Trash

## Seventh Project – File Downloader (Intermediate level Capstone)



## Eighth Project – Quiz bank API

## Nineth Project – Registration form (Web)

## Tenth Project – Ecommerce Inventory manager (Advanced Level Capstone



# Prepare To Code

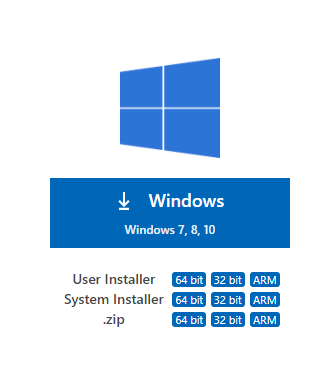
## Install python

Open a browser window and type in [python.org](https://python.org) in the URL field. This is where the python download files are located. You can also click on the following image if looking at the eBook. We are currently working with version 3.9.1, but if you have an older version you may need to install helper programs (such as pip) separately.

[](https://python.org/)

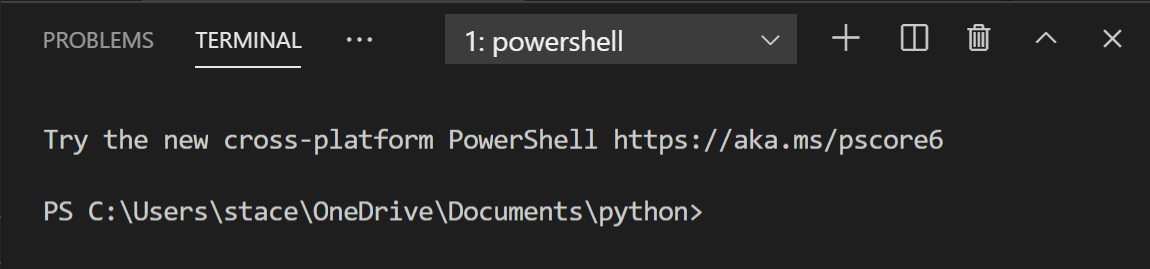
## Install VS code editor

To get started pulling up code from GitHub, you will need to download the Visual Studio Code editor. Go to <https://code.visualstudio.com/download> to install the latest version for your computer. This is the most popular editor that is used with many different programming languages, and we will be looking at it from the perspective of windows 10 64-bit installation.

[](https://code.visualstudio.com/download)

## Install Gitbash

The GitBash tool allows you to install other packages in python and put in commands pertaining to your version control account on GitHub. This is a terminal service that is very handy for developers. Within VS Code editor, you will see at the bottom of the screen, a tab that is called the terminal.



You will be able manage your projects here. Go to <https://git-scm.com/downloads> for the latest version. You can also click on the following image to access this download file from the eBook. You may see this image on the website to begin the process.

[](https://git-scm.com/downloads)

## Sign up for github

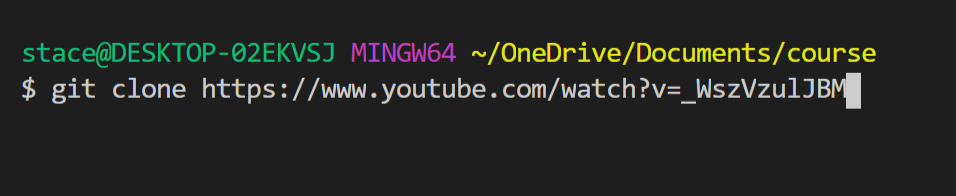
To access and manage the base code included in this course, you will need a GitHub account. Go to github.com to begin the process. You can also click on the following image to access the website from the eBook.

## Clone repository

Go to GitHub and clone the Popcode repository. You will not even have to copy and paste code from a YouTube tutorial or stack overflow article. This is a sure-fire way to get your code into your code editor without a hassle.

At the terminal’s command line or in GitBash, you will have to enter the following command:

* Git clone <https://github.com/staceyloulouse/popcode.git>



## Take notes during code review

It is important that you capture, in your own words, how the code is structured. When I talk about a specific syntax or coding structure, it is a good time to open up a tab in VS Code and make note of what you have gleaned from the narrative.

## Understand the problem

You can never get to the finish line if you do not know the race route. Your job, as a developer will be to design the turn-to-turn blueprint with markers to the finish point. Before you can get to this point in development, you must know what the problem or definition of the solution is needed. Understanding the problem is simply knowing what the user(s) of the program ultimately want or need. A freelance developer’s greatest job is to be good at taking goals of a client and turning it into a problem statement. When working on an original problem, there is usually a defining moment or challenge that is experienced.

## Imagine the end result

Being able to imagine the result is a great skill to have. Having the end in mind will allow you to build a prototype of the deployable project. You will see a prototype image of each project, but every one of those prototypes began with a hand-drawn sketch of what the final application would consist of or look like. You can start this process at the beginning of any project development instead of diving deep into coding first.

# Register & Attend

## Self-paced option

You do not need to register or attend if taking the self-paced option. Going as fast or slow as you want has its advantages. You will not be able to as the instructor questions, but you can take notes on the code reviews, post comments on them for feedback. Taking the live session series requires addition payment beyond the cost of the book, so you will save money this way. If you want to finish in less or more time that the regular pace of the live session series, then you are free to do that. Feel free to skip to first project. Thank you for your purchase.

## Thank you for your purchase

You could have gotten instruction on python from many different venues, so your purchase is very much appreciated.

## Project Code Reviews

In addition to live sessions, there will be an opportunity to follow the links in the book to go to the code review for each project. Please take the time to watch the videos before or after the sessions to reinforce learning.

## Limited Class Size

Each class can have no more than 25 people in. Limited class size allows students to actively participate and get their questions answered.

## Zoom Meeting ID and password

# MEETING ID: 317-920-0956

# PASSWORD: 058021

## Links to Code Review Videos

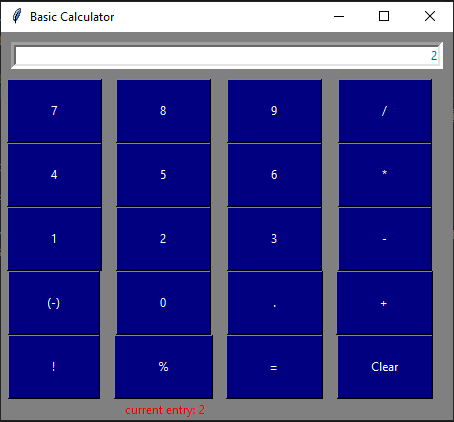
# YouTube channel: [POPcode maker](https://www.youtube.com/channel/UC5qzIC9P3ILDb7Pd3pWwuDQ)

## Questions? Email:

# stacey@popcodemaker.com

# First Project

## Basic calculator



# Basic Calculator

## Goal: provide answer to basic arithmetic operations, including percent, factorial, and negate

## Uses tkinter module

## Uses function definitions

## Uses lambda expression

## Has a status message area for announcements

# First Project – Scope

## Use Case

## Desktop Application

## Serves as an alternate to calculator

## User Stories

## Screen has arithmetic function operations

## Screen has 10 digits for input

## Screen displays status message for error or updated information

## Screen shows results of arithmetic operations

## Screen has factorial, negate, and percent feature

# Project Resources

## VS Code Editor

## github clone

## Python 3 software

## pip

## Tkinter module

## NSIS software

## pyinstaller

# Review Video

At this point, you should watch the video for the first project. Here is the link:

<https://youtu.be/S0VmJniLV48>

[Logo, company name

Description automatically generated](https://youtu.be/S0VmJniLV48)

# First Project – Code Review

## We used the tkinter module by using the keyword “import”

## We used an alias name for tkinter by using the keyword “as”

## We accomplished variable assignments with an equal sign (=)

## We refer to methods/functions by using a dot (.) between the variable and function name

## We use a colon (:) at the end of function definitions (with the keyword **DEF)**, try/except blocks, and while loops

## VS code automatically indents when the colon is put after these special statements (def, try, except, while)

## The colon and indentation is required in other python statements as well (if, for)

# **Project code**

1. import tkinter as tk
2. from tkinter import Button
3. from tkinter import StringVar
4. from tkinter import Entry
5. from tkinter import Label
6. # setup the frame of the interface and give it a title and background color
7. root = tk.Tk()
8. root.title('Basic Calculator')
9. root.configure(background='light blue')
10. # operator string
11. operator = ''
12. textInput = StringVar()
13. statusText = StringVar()
14. # setup calculator environment
15. e = Entry(root, fg = 'green', textvariable= textInput,justify='right',width = 70, borderwidth = 5)
16. e.grid(row = 0, column = 0, columnspan = 5, padx=20,pady=20)
17. # reaction to clicking the numbers and operator buttons
18. def click\_action(input):
19. global operator
20. global statusText
21. operator += str(input)
22. textInput.set(operator)
23. statusText.set('current entry: ' + operator)
25. # preview area function
26. def updateStatus():
27. global operator
28. global statusText
29. statusText = statusText + operator
30. statusText.set(statusText)
31. # clear button function
32. def actionClear():
33. global operator
34. global statusText
35. statusText.set(operator + ' cleared')
36. operator = ''
37. textInput.set('0')
38. # equal button function
39. def actionEqual():
40. global operator
41. global statusText
42. answer = ''
43. try:
44. answer = str(eval(operator))
45. textInput.set(answer)
46. statusText.set(operator + '= '+ answer)
47. except SyntaxError as y:
48. statusText.set('Error')
49. print(y.args)
51. # negate button function
52. def actionNegate():
53. global operator
54. operator = '-('+operator+')'
55. textInput.set(operator)
56. # percent function
57. def actionPercent():
58. global operator
59. try:
60. answer = str(eval(operator)/100)
61. textInput.set(answer)
62. statusText.set('('+ operator + ')%= '+ answer)
63. operator = answer
64. except SyntaxError as y:
65. statusText.set('Error')
66. print(y.args)
67. def actionFactorial():
68. global operator
69. try:
70. answer = 1
71. base = int(str(eval(operator)))
72. statusText.set('current entry: '+operator)
73. while base>1:
74. answer\*=base
75. base-=1
76. textInput.set(answer)
77. statusText.set('('+ operator + ')!= '+ str(answer))
78. except SyntaxError:
79. statusText.set('Error')
80. except ValueError:
81. statusText.set('Only Integers Expected')
82. #define buttons in top row (7,8,9,/)=================================================
83. button7 = Button(root, text = "7", command=lambda: click\_action(7),padx=40,pady=20, bg='navy', fg='white')
84. button7.grid(row=2, column=0)
85. button8 = Button(root, text = "8", command=lambda: click\_action(8),padx=40,pady=20, bg='navy', fg='white')
86. button8.grid(row=2, column=1)
87. button9 = Button(root, text = "9", command=lambda: click\_action(9),padx=40,pady=20, bg='navy', fg='white')
88. button9.grid(row=2, column=2)
89. buttonDivide = Button(root, text = "/", command = lambda: click\_action("/"), padx=40,pady=20, bg='navy', fg='white')
90. buttonDivide.grid(row=2, column=3)
91. #Second row of buttons (4, 5,6,\*)===================================================
92. button4 = Button(root, text = "4", command=lambda: click\_action(4),padx=40,pady=20, bg='navy', fg='white')
93. button4.grid(row=3, column=0)
94. button5 = Button(root, text = "5", command=lambda: click\_action(5),padx=40,pady=20, bg='navy', fg='white')
95. button5.grid(row=3,column=1)
96. button6 = Button(root, text = "6", command=lambda: click\_action(6),padx=40,pady=20, bg='navy', fg='white')
97. button6.grid(row=3, column=2)
98. buttonMultiply = Button(root, text = "\*", command=lambda: click\_action('\*'),padx=40,pady=20, bg='navy', fg='white')
99. buttonMultiply.grid(row=3, column=3)
100. #Third row of buttons (1,2,3,-)=======================================================
101. button1 = Button(root, text = "1", command=lambda: click\_action(1),padx=40,pady=20, bg='navy', fg='white')
102. button1.grid(row=4, column=0)
103. button2 = Button(root, text = "2", command=lambda: click\_action(2),padx=40,pady=20, bg='navy', fg='white')
104. button2.grid(row=4, column=1)
105. button3 = Button(root, text = "3", command=lambda: click\_action(3),padx=40,pady=20, bg='navy', fg='white')
106. button3.grid(row=4, column=2)
107. buttonSubtract = Button(root, text = "-", command=lambda: click\_action('-'),padx=40,pady=20, bg='navy', fg='white')
108. buttonSubtract.grid(row=4, column=3)
109. #Fourth row of buttons ((-),0,+,.)=======================================================
110. buttonNegate = Button(root, text = "(-)", command = lambda: actionNegate(), padx=35,pady=20, bg='navy', fg='white')
111. buttonNegate.grid(row=5, column=0)
112. button0 = Button(root, text = "0", command=lambda: click\_action(0),padx=40,pady=20, bg='navy', fg='white')
113. button0.grid(row=5, column=1)
114. buttonDecimal = Button(root, text = ".", command=lambda: click\_action('.'),padx=42,pady=20, bg='navy', fg='white')
115. buttonDecimal.grid(row=5, column=2)
116. buttonAdd = Button(root, text = "+", command=lambda: click\_action('+'),padx=40,pady=20, bg='navy', fg='white')
117. buttonAdd.grid(row=5, column=3)
118. #Fifth row of buttons (x,y,,,%,nPr)==========================================================
119. buttonFactorial = Button(root, text = "!", command = lambda: actionFactorial(),padx=40,pady=20, bg='navy', fg='white')
120. buttonFactorial.grid(row=6, column=0)
121. buttonPercent = Button(root, text = "%", command = lambda: actionPercent(), padx=40,pady=20, bg='navy', fg='white')
122. buttonPercent.grid(row=6,column=1)
123. buttonEqual = Button(root, text = "=", command = lambda: actionEqual(), padx=40,pady=20, bg='navy', fg='white')
124. buttonEqual.grid(row=6, column=2)
125. buttonClear = Button(root, text = "Clear", command = lambda: actionClear(),padx=30,pady=20, bg='navy', fg='white')
126. buttonClear.grid(row=6, column=3)
127. #status message area
128. statusMessage = Label(root, fg = 'white', textvariable= statusText, bg='grey')
129. statusMessage.grid(row=9,column=0,columnspan=3)
130. # repeat root display until window closes
131. root.mainloop()

# First Project – Exercises

## Change the title to **“Python Calculator”**

## Change the entry widget to a **label widget** for displaying the answer/input to the user

## Change the background color **(bg)** of the number buttons.

## Change the negate function to a reciprocal function **(1/x)**

## The reciprocal function will have a **button** associated with it. update the function with the appropriate logic.

# First Assessment

In the live class, an assessment will be done during the QUESTION-AND-ANSWER period. For self-paced learners, the assessment questions are at the end of the eBook.

## [JUmp to The first assessment](#_First_assessment)

# Collaboration Activity

## Sign up for GitHub and repl.it

## Start a repository

## Clone your repo on your local drive where your code goes

## Add the calculator app to your local drive and push to your repo

## Talk to someone about your project or share your repl.it

# First Project – Deployment Plan

## To make this desktop application executable you will need to do the following:

## Make sure you have pip installed (check using the command: pip --version)

## Install pyinstaller using the command: pip install pyinstaller

## Run the command: pyinstaller –onefile –w app\_name.py

## Download NSIS from [**https://sourceforge.net/projects/nsis/**](https://sourceforge.net/projects/nsis/)

## After installing the software, zip the project folder and click on installer based on zip file

# Enhance and Make it Yours!

## Think of how you can make the calculator better.

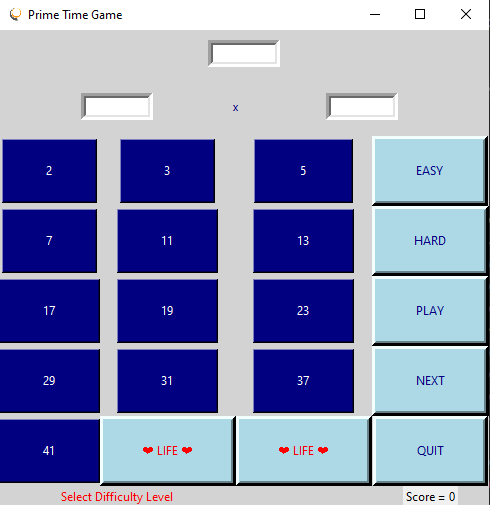
## Is there a feature that you can put on the calculator?

## Brand your idea for the calculator to add to your portfolio or put your repo on your resume.

## Be sure to zip and re-run the executable installer in NSIS for your version of the calculator

# Second Project

## Prime Time Game



# Prime Time Game

## Goal: select two FACTORS for the given composite

## Uses tkinter module

## Uses random module

## Uses unicode characters

## Keep score, 100 points for correct answers until 5000, then move to hard level, 500 points each

## Lose a life (3) for incorrect answers

# Second Project – Scope

## Use Case

## Fifth grade class learns about multiplication and division

## Teacher needs an aid to help them learn about factors, primes, and composite numbers

## Desktop application for school PCs and loaner laptops

## User Stories

## Goal: provide an educational game for students to learn

## The game responds to inaccurate user input

## The game has a message area

## The game has a scoring feature

## The screen has playing levels

## The game has sound effects

# Project Resources

## VS Code Editor

## gitbash

## github clone

## Python 3 software

## pip

## Tkinter module

## NSIS software

## pyinstaller

# Review Video

At this point, you should watch the video for the second project. Here is the link:

<https://youtu.be/m6L2f1gVMQ8>

[Logo, company name

Description automatically generated](https://youtu.be/m6L2f1gVMQ8)

# Second Project – Code Review

## We used the random module to display a random composite number

## We used if statements to control the flow of the application

## We used pip to install pillow module for the ImageTk and Image features

## We changed the state of a button using the config method in tkinter

## We used the playsound module to play a sound file

## VS code automatically changes all occurrences of a variable when you use the hot key: Ctrl+F2

## The strings in python include special characters using escape sequence “\u” and then the Unicode number

# **Project code**

import tkinter as tk

from tkinter import Button

from tkinter import StringVar

from tkinter import IntVar

from tkinter import Entry

from tkinter import Label

import random

import time

from PIL import ImageTk, Image

from playsound import playsound

#setup the frame of the interface and give it a title

root = tk.Tk()

root.title("Prime Time Game")

root.iconbitmap('C:\\Users\\stace\\OneDrive\\Documents\\python\\1PrimeTime\\iconImage.ico')

root.configure(background='light grey')

#variable definitions

operator = ''

textInput = StringVar()

f1Input = StringVar()

f2Input = StringVar()

score = 0

statusText = StringVar(root,'Select Difficulty Level')

scoreText = StringVar(root,'Score = '+ str(score))

life1=True

life2=True

zap\_life=2

primes=[2,3,5,7,11]

increase\_by=100

#setup prime time game environment

e = Entry(root, fg = 'navy', textvariable= textInput, justify='right', width = 10, borderwidth = 5)

e.grid(row = 0, column = 0, columnspan = 5, padx=10,pady=10)

f1 = Entry(root, fg = 'navy', textvariable= f1Input,justify='right',width = 10,borderwidth = 5)

f1.grid(row = 1, column = 0, columnspan = 2, padx=10,pady=10)

f2 = Entry(root, fg = 'navy', textvariable= f2Input, justify='right', width = 10, borderwidth = 5)

f2.grid(row = 1, column = 2, columnspan = 3, padx=10,pady=10)

signLabel = Label(root, fg='navy', text='x', justify='center', bg='light grey', padx=10, pady=10)

signLabel.grid(row=1,column=1,columnspan=2, padx=10, pady=10)

#reaction to clicking the number buttons

def numClick(number):

    global operator

    global statusText

    global f1Input, f2Input

    statusText.set('current entry: ' + str(number))

    if f1.get() == '':

        operator = str(number)

        f1Input.set(operator)

    else:

        operator+='\*'+ str(number)

        f2Input.set(number)

        checkEntry()

# checking to see if the selections are correct

def checkEntry():

    global f1Input, f2Input, operator

    global life1,life2

    try:

        answer = str(eval(operator))

        statusText.set(operator + '= '+ answer)

        if str(answer) == textInput.get():

            statusText.set('Excellent! Click Next.')

            scoreUp()

            scoreText.set('Score: '+ str(score))

            #play chime or applause

            playsound('C:\\Users\\stace\\OneDrive\\Documents\\python\\1PrimeTime\\TaDaSound.mp3')

        else:

            statusText.set('Wrong!')

            life\_loss()

            #play chime or smashing

            playsound('C:\\Users\\stace\\OneDrive\\Documents\\python\\1PrimeTime\\Smashing.mp3')

    except SyntaxError as y:

        statusText.set('Error')

        print(y.args)

# changing the difficulty level

def setLevel(level):

    global primes

    global increase\_by

    if level == 'EASY':

        increase\_by=100

        primes = [2,3,5,7,11,13,17]

        statusText.set('Level EASY, Let\'s PLAY!')

    else:

        increase\_by=500

        statusText.set('Level: HARD, You\'re Brave!')

        primes = [2,3,5,7,11,13,17,19,23,29,31,37,41]

#lose a life

def life\_loss():

    global zap\_life

    if zap\_life>1:

        buttonLife2.destroy()

        zap\_life-=1

    elif zap\_life>0:

        buttonLife1.destroy()

        zap\_life-=1

    else:

        gameOver()

# action for the play button

def play():

    #Start or restart the game

    global operator

    global score

    global primes

    statusText.set('New Game. Let\'s Go!')

    score=0

    scoreText.set('Score: '+ str(score))

    buttonLife1.config(text = '\u2764 LIFE \u2764')

    buttonLife2.config(text = '\u2764 LIFE \u2764')

    # clear the display fields

    operator = ''

    textInput.set('')

    f1Input.set('')

    f2Input.set('')

    actionNext()

    buttonPlay.config(state='disabled')

#destroy the prime number buttons when the game is over

def gameOver():

    for blink in range(1,101):

        statusText.set('GAME OVER! Click Quit or Play')

        textInput.set('Game Over')

        if blink%2==0:

            showScore.config(fg='red')

            statusMessage.config(fg = 'red')

        else:

            showScore.config(fg='white')

            statusMessage.config(fg = 'white')

        statusText.set('Click NEXT or Quit!')

# increase the player's score by the increment amount

def scoreUp():

    global score

    global increase\_by

    score += increase\_by

    scoreText.set('Score: '+ str(score))

    if score>5000:

        statusText.set('Leveled Up to HARD!')

        setLevel('HARD')

    elif score>7000:

        increase\_by=1000

#next button function

def actionNext():

    global operator

    global statusText

    global primes

    statusText.set(operator + ' cleared')

    operator = ''

    textInput.set('')

    f1Input.set('')

    f2Input.set('')

    # display a random composite number

    n=len(primes)-1

    num1=primes[random.randint(0,n)]

    num2=primes[random.randint(0,n)]

    textInput.set(num1\*num2)

# buttons

#define buttons in top row (2,3,5,EASY)=================================================

button2 = Button(root, text = "2", command=lambda: numClick(2),padx=40,pady=20, bg='navy', fg='white')

button2.grid(row=2, column=0)

button3 = Button(root, text = "3", command=lambda: numClick(3),padx=40,pady=20, bg='navy', fg='white')

button3.grid(row=2, column=1)

button5 = Button(root, text = "5", command=lambda: numClick(5),padx=42,pady=20, bg='navy', fg='white')

button5.grid(row=2, column=2)

# Second row of buttons (7,11,13,HARD)===================================================

button7 = Button(root, text = "7", command=lambda: numClick(7),padx=40,pady=20, bg='navy', fg='white')

button7.grid(row=3, column=0)

button11 =Button(root, text = "11", command=lambda: numClick(11),padx=40,pady=20, bg='navy', fg='white')

button11.grid(row=3,column=1)

button13 =Button(root, text = "13", command=lambda: numClick(13),padx=40,pady=20, bg='navy', fg='white')

button13.grid(row=3, column=2)

#Third row of buttons (17,19,23,PLAY)=======================================================

button17 =Button(root, text = "17", command=lambda: numClick(17),padx=40,pady=20, bg='navy', fg='white')

button17.grid(row=4, column=0)

button19 =Button(root, text = "19", command=lambda: numClick(19),padx=40,pady=20, bg='navy', fg='white')

button19.grid(row=4, column=1)

button23 = Button(root, text = "23", command=lambda: numClick(23),padx=40,pady=20, bg='navy', fg='white')

button23.grid(row=4, column=2)

#Fourth row of buttons (29,31,37,NEXT)=======================================================

button29 = Button(root, text = "29", command = lambda: numClick(29), padx=40,pady=20, bg='navy', fg='white')

button29.grid(row=5, column=0)

button31 = Button(root, text = "31", command=lambda: numClick(31),padx=40,pady=20, bg='navy', fg='white')

button31.grid(row=5, column=1)

button37 = Button(root, text = "37", command=lambda: numClick(37),padx=40,pady=20, bg='navy', fg='white')

button37.grid(row=5, column=2)

#Fifth row of buttons (41,Life,Life,QUIT)==========================================================

button41 = Button(root, text = "41", command = lambda: numClick(41), padx=40,pady=20, bg='navy', fg='white')

button41.grid(row=6, column=0)

# navigation buttons on fourth column (EASY, HARD, PLAY, NEXT, QUIT)

buttonEasy = Button(root, text = "EASY", command = lambda: setLevel('EASY'), padx=37,pady=20, bg='light blue', fg='navy',borderwidth = 5)

buttonEasy.grid(row=2, column=3)

buttonHard = Button(root, text = "HARD", command=lambda: setLevel('HARD'),padx=35,pady=20, bg='light blue', fg='navy',borderwidth = 5)

buttonHard.grid(row=3, column=3)

buttonPlay = Button(root, text = "PLAY", command=play,padx=37,pady=20, bg='light blue', fg='navy',borderwidth = 5)

buttonPlay.grid(row=4, column=3)

buttonNext = Button(root, text = "NEXT", command=lambda: actionNext(),padx=37,pady=20, bg='light blue', fg='navy',borderwidth = 5)

buttonNext.grid(row=5, column=3)

buttonQuit = Button(root, text = "QUIT", command = root.quit,padx=37,pady=20, bg='light blue', fg='navy',borderwidth = 5)

buttonQuit.grid(row=6, column=3)

global buttonLife1

buttonLife1 = Button(root, text = "\u2764 LIFE \u2764" ,padx=35,pady=20, bg='light blue', fg='red',borderwidth = 5)

buttonLife1.grid(row=6, column=1)

global buttonLife2

buttonLife2 = Button(root, text = "\u2764 LIFE \u2764", padx=35,pady=20, bg='light blue', fg='red',borderwidth = 5)

buttonLife2.grid(row=6,column=2)

#status message area

statusMessage = Label(root, fg = 'navy', textvariable= statusText, bg='light grey')

statusMessage.grid(row=9,column=0,columnspan=2)

# label for score report

showScore = Label(root, textvariable = scoreText)

showScore.grid(row=9, column=3)

# repeat root display until window closes

root.mainloop()

# Second Project – Exercises

## Change the entry widget to a **label widget** for displaying the random composite number

## Change the life buttons to a Different Unicode Character

## Change the background color **(bg)** of the Screen when the game is over

## Change the next button to a start button and adjust the play method to loop until the game is over

## replace the start button with a medium Level button

# Second Assessment

In the live class, an assessment will be done during the QUESTION-AND-ANSWER period. For self-paced learners, the assessment questions are at the end of the eBook.

## [JUmp to The Second assessment](#_Second_assessment)

# Collaboration Activity

## Sign into repl.it and add your GitHub repo that created

## Add Second project files to your drive and push the edited version to GitHub

## Share your link in repl.it or your github.com/profile\_name with Stacey@popcodemaker.com for feedback

## Talk to someone about your project to articulate the progress you have made.

# Second Project – Deployment Plan

## Use pyinstaller to make onefile version of the project

## Zip the files into a folder

## Use NSIS to create your executable program

## Share your project!

# Enhance and Make it Yours!

## Think of how you can make the game better.

## Is there a feature that you can put on the game?

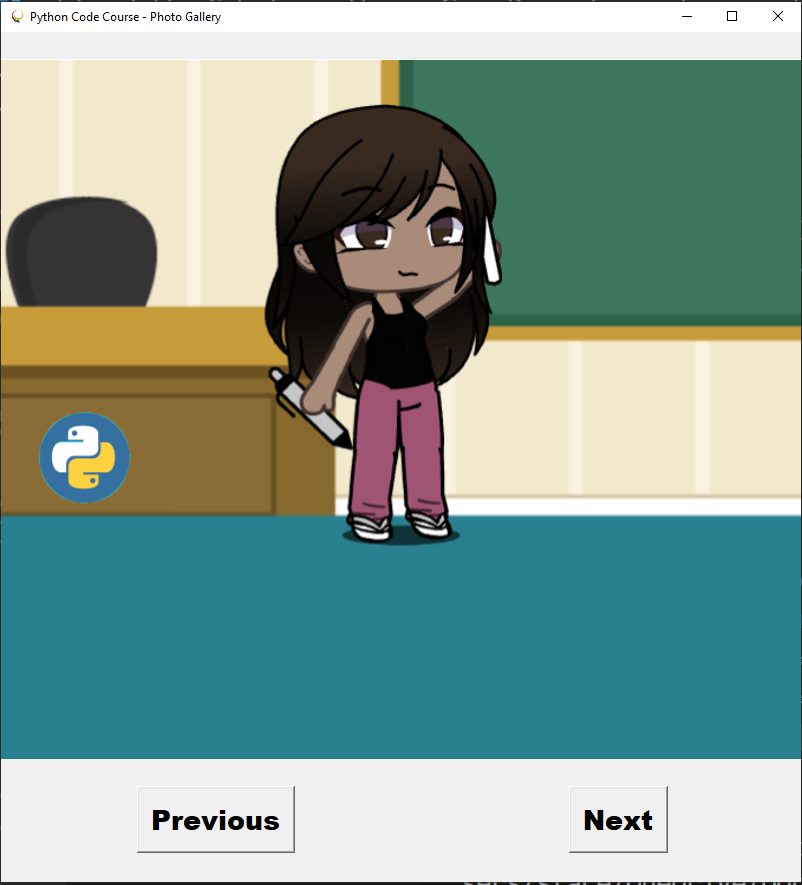
## Will you change the layout?

## Can you add a scoresheet with the highest score?

## Brand your idea for the app and add to your portfolio.

# Third Project

## Photo Gallery



# Photo Gallery App

## Goal: provide scroll through image feature

## Uses tkinter module

## Uses file handling

## Uses list data structures

## Has a forward scroll and backward scroll option

# Third Project – Scope

## Use Case

## Grandma wants to view all downloaded photos in her PC

## Desktop Application

## Consolidate image files into a photo album

## User Stories

## The screen should shows all photos individually

## The screen should provide a way to make a selection to progress to next photo

## The app contains bank of photo files

## The app provides a selection for the next or previous photo

# Project Resources

## VS Code Editor

## github clone

## Python 3 software

## PIL

## Tkinter module

## NSIS software

## pyinstaller

# Review Video

At this point, you should watch the video for the third project. Here is the link:

<https://youtu.be/TOUy8Rlu7xM>

[Logo, company name

Description automatically generated](https://youtu.be/TOUy8Rlu7xM)

# Third Project – Code Review

## We introduce a list by declaring a variable and using square brackets **[]**.

## We introduce **for** loops to iterate through items in a list

## A file can be accessed after using the **open** function with its file name.

## Adding an element to a list can be done with the **append** function

## A function call uses the parenthesis without arguments or with arguments (if defined).

## You can return a value to the calling statement using the keyword **“return”**

## You can add an image to a label using the **“image”** attribute

# **Project code**

from tkinter import \*

from PIL import ImageTk, Image

#main screen

screen = Tk()

screen.title('Python Code Course - Photo Gallery')

screen.iconbitmap('C:\\Users\\stace\\OneDrive\\Documents\\python\\2quizBit\\iconImage.ico')

screen.geometry('800x850')

# initialize variables

images = []

count = 0

def load\_photos():

    global images

    album = open('C:\\Users\\stace\\OneDrive\\Documents\\python\\3mediaViewer\\photos.txt')

    next\_photo = album.readlines()

    next\_photo = next\_photo[0].split(',')

    for item in next\_photo:

        images.append(ImageTk.PhotoImage(Image.open(item)))

    album.close()

def view\_next():

    global count

    global images

    if count<len(images)-1:

        count+=1

        photo\_label.config(image = images[count])

    else:

        count = 0

        photo\_label.config(image = images[count])

    return images

def view\_previous():

    global count

    global images

    if count>0:

        count-=1

        photo\_label.config(image = images[count])

    else:

        count = len(images)-1

        photo\_label.config(image = images[count])

    return images

# function call

load\_photos()

#define widgets for the screen

photo\_label = Label(image = images[count], width = 800, height = 750)

photo\_label.grid(row = 0, column = 0, columnspan = 2)

next\_button = Button(screen, text = 'Next', font = ('Arial black', 20), command = view\_next)

next\_button.grid(row = 1, column = 1)

previous\_button = Button(screen, text = 'Previous', font = ('Arial black', 20), command = view\_previous)

previous\_button.grid(row = 1, column = 0)

screen.mainloop()

# Third Project – Exercises

## Include an add photo button for accepting new images for the album

## Update the load photos function so that the new images can be read

## Change the background color **(bg)** of the Screen

## Add an entry widget so the user can name their album

## Include a delete button and remove the image from the list and photo.txt file

# Third Assessment

In the live class, an assessment will be done during the QUESTION-AND-ANSWER period. For self-paced learners, the assessment questions are at the end of the eBook.

## [JUmp to The third assessment](#_Third_assessment)

# Collaboration activity

## Push your file changes to github

## Import your github files into repl.it

## Share your repl.it link with your instructor for feedback

## Talk to someone about your project

# Third Project – Deployment Plan

## Use pyinstaller to make onefile version of the project

## Zip the files into a folder

## Use NSIS to create your executable program

## Share your project!

# Enhance and Make it Yours!

## Think of how you can make the photo gallery better.

## Is there a feature that you can put on the gallery?

## Brand your idea for the photo gallery to add to your portfolio.

# Fourth Project

## Widgets



# Widgets

## Goal: provide capability to solve widget declaration and implementation

## Uses tkinter module

## Uses SQLITE3 module

## Uses function definitions

## Uses lambda expression

## Uses database management features

## Has label, entry, button, slider, frame, checkbox, drop down menu, and radio button widgets

# Fourth Project – Scope

## Use Case

## Students need to learn all about widgets in tkinter module

## Desktop Application

## Consolidate common widgets into a module for ease of access

## User Stories

## The screen should shows all types of questions similar to certification

## The screen should provide a way to make a selection or input of the correct answer

## The screen contains animation

## The app contains bank of questions in a file

# Project Resources

## VS Code Editor

## GitHub clone

## Python 3 software

## PIP

## Tkinter module

## NSIS software

## pyinstaller

# Review Video

At this point, you should watch the video for the fourth project. Here is the link:

<https://youtu.be/Zr5EUh2XDr0>

[Logo, company name

Description automatically generated](https://youtu.be/Zr5EUh2XDr0)

# Fourth Project – Code Review

## We use database tool that is A built-in module

## We used the **execute()** method to render sql commands

## We used the **fetchall()** command to get all the output from the query

## We used dictionaries with key/value pairs, to define the columns in the table of the database

## **OptionMenu** is the widget that allows you to select from a drop down box.

## We used a **messagebox()** to show a pop-up message to the user.

## The **pack()** layout lets us have a generalLy ordered placement of items on a screen

## We used new widgets for this module such as the **checkbox**, **radiobutton**, and sliders (**SCALE()** widget)

# **Project code**

import sqlite3

import tkinter

from tkinter import \*

from tkinter import messagebox

from tkinter import filedialog

from PIL import ImageTk, Image

from tkinter import Scale

main\_screen = tkinter.Tk()

main\_screen.title('Widgets Tutorial')

main\_screen.iconbitmap('C:\\Users\\stace\\OneDrive\\Documents\\python\\2quizBit\\iconImage.ico')

main\_screen.configure(background = 'grey')

student\_options = []

sql\_db = sqlite3.connect('users.db')

control = sql\_db.cursor()

#create a table within the database using the control cursor

control.execute('SELECT oid,\* FROM students')

entries = control.fetchall()

for entry in entries:

    student\_options.append(entry)

sql\_db.commit()

sql\_db.close()

student\_clicked = StringVar(main\_screen, student\_options[0])

submissions = []

lunch = ['No cheese', 'No peppers', 'No mushrooms', 'and No tomatoes.']

rate\_num = StringVar()

year\_num = StringVar()

has\_cheese = IntVar()

has\_peppers = IntVar()

has\_mushrooms = IntVar()

has\_tomatoes = IntVar()

toppings = StringVar()

entree\_clicked = StringVar(main\_screen, 'Pizza')

student\_clicked = StringVar(main\_screen, student\_options[0])

photo = ImageTk.PhotoImage(Image.open('C:\\Users\\stace\\OneDrive\\Documents\\python\\pylogo.png'))

toggle = True

def clickAction():

    global toggle

    global start\_button

    if toggle:

        start\_button.config(bg = 'black', fg = 'white', text = 'Thank you!')

        toggle = False

    else:

        start\_button.config(bg = 'white', fg = 'black')

        toggle = True

def submit\_action():

    #connect to database

    sql\_db = sqlite3.connect('users.db')

    control = sql\_db.cursor()

    control.execute('INSERT INTO students VALUES (:name, :email)',

            {

                'name':user\_entry.get(),

                'email':email\_entry.get()

            })

    #update the database and close it

    sql\_db.commit()

    sql\_db.close()

    submit\_button.config(text = 'Success!', bg = 'light blue', fg = 'white')

    submissions.append({user\_entry.get(): email\_entry.get()})

    submit\_text = 'Hi, ' + user\_entry.get()+' '+email\_entry.get()

    Label(main\_screen, text = submit\_text, font = ('Arial black', 22)).grid(row = 3, column = 1, pady = 10)

    user\_entry.delete(0,END)

    email\_entry.delete(0,END)

#show results of student entries into the database

def show\_entries():

    #connect to database

    sql\_db = sqlite3.connect('users.db')

    control = sql\_db.cursor()

    control.execute('SELECT oid,\* FROM students')

    #use fetchall, fetchmany(#howmany), or fetchone

    entries = control.fetchall()

    entry\_string = ''

    for entry in entries:

        entry\_string+=str(entry)+'\n'

    Label(main\_screen, text = entry\_string).grid(row = 14, column = 1)

    #update the database and close it

    sql\_db.commit()

    sql\_db.close()

def delete\_entry():

    #connect to database

    sql\_db = sqlite3.connect('users.db')

    control = sql\_db.cursor()

    id\_num = student\_clicked.get().split(',')

    id\_num = id\_num[0][1:]

    control.execute('DELETE FROM students WHERE oid = ' + id\_num)

    #update the database and close it

    sql\_db.commit()

    sql\_db.close()

# update an email entry for a student

def update\_entry(num):

    update\_screen = tkinter.Tk()

    update\_screen.title('Update Student Email')

    update\_screen.configure(background = 'pink')

    #connect to database

    sql\_db = sqlite3.connect('users.db')

    control = sql\_db.cursor()

    control.execute('''

        SELECT oid,\* FROM students

    ''')

    emails = control.fetchall()

    email\_clicked = StringVar(update\_screen, emails[0])

    # drop down menu of students

    OptionMenu(update\_screen, email\_clicked, \*emails).pack()

    # entry label and field for email input

    Label(update\_screen, text = 'new email').pack()

    new\_email = Entry(update\_screen, borderwidth = 5 )

    new\_email.pack()

    control.execute('UPDATE students SET email = :email WHERE oid = ' + str(num), {'email':new\_email.get()})

    #update the database and close it

    sql\_db.commit()

    sql\_db.close()

#show result of checking radio button with messagebox

#other messagebox methods --> showwarning, showerror, askquestion, askokcancel, askyesno

def check(value):

    if value == 'quotes':

        messagebox.showinfo('Result', 'That\'s, correct!')

    else:

        messagebox.showinfo('Result', 'That\'s, not right!')

#use a dialog box to get the file name of the photo to use

def open\_image\_file():

    global picture

    open\_frame\_icon = filedialog.askopenfilename(initialdir = '/', title = 'Choose a photo')

    open\_label = Label(open\_frame, text = open\_frame\_icon)

    open\_label.pack()

    picture = ImageTk.PhotoImage(Image.open(open\_frame\_icon))

    open\_icon = Label(open\_frame, image = picture, width = 200, height = 200)

    open\_icon.pack\_forget()

    open\_icon.pack()

# function to assign a value to rate\_num based on the slider

def get\_rating():

    global rate\_num

    rate\_num.set(wide.get())

def get\_year():

    global year\_num

    year\_num.set(tall.get())

def get\_lunch():

    global lunch

    order = 'Your lunch comes with: Sauce, '

    if has\_cheese.get()==1:

        order += 'Cheese, '

    else:

        order += lunch[0] + ', '

    if has\_peppers.get()==1:

        order += 'Peppers, '

    else:

        order += lunch[1] + ', '

    if has\_mushrooms.get()==1:

        order += 'Mushrooms, '

    else:

        order += lunch[2] + ', '

    if has\_tomatoes.get()==1:

        order += 'and Tomatoes. '

    else:

        order += lunch[3]

    lunch\_label = Label(main\_screen, textvariable = toppings)

    lunch\_label.grid(row = 12, column = 0)

    toppings.set(order)

#label widgets

icon\_label = Label(image = photo)

icon\_label.grid(row = 13, column = 0)

welcome\_label = Label(main\_screen, text = 'Welcome to Widgets!', bg = 'grey',font = ('Arial black', 22) )

welcome\_label.grid(row = 0, column = 1, columnspan = 2)

name\_label = Label(main\_screen, text = 'Enter your name:', bg = 'grey', font = ('Arial black', 22))

name\_label.grid(row = 1, column = 0)

email\_label = Label(main\_screen, text = 'Enter your email:', bg = 'grey', font = ('Arial black', 22))

email\_label.grid(row = 1, column = 2)

#define frames

open\_frame = LabelFrame(main\_screen, text = 'Select an image', padx=20, pady=20)

open\_frame.grid(row = 4,column = 0)

option\_frame = LabelFrame(main\_screen, text = 'What symbol is used to define strings?', padx=50, pady = 50)

option\_frame.grid(row = 4, column = 1)

#button widget

start\_button = Button(main\_screen,text = 'Finish', bg = 'yellow', command = clickAction,font = ('Arial black', 22), fg = 'light blue')

start\_button.grid(row = 4, column = 4)

submit\_button = Button(main\_screen, text = 'Submit', bg = 'yellow', command = submit\_action,font = ('Arial black', 22), fg = 'light blue')

submit\_button.grid(row = 1, column = 4, columnspan = 4, pady = 5, padx = 10)

exit\_button = Button(main\_screen, text = 'Exit', bg = 'yellow', command = main\_screen.quit, font = ('Arial black', 22), fg = 'light blue')

exit\_button.grid(row = 13, column = 4)

open\_button = Button(open\_frame, text = 'Choose photo', bg = 'red', command = open\_image\_file)

open\_button.pack()

delete\_button = Button(main\_screen, text = 'Delete Student', bg = 'yellow', command = delete\_entry, font = ('Arial black', 22), fg = 'light blue')

delete\_button.grid(row = 7, column = 4)

update\_button = Button(main\_screen, text = 'Update Student', bg = 'yellow', command = update\_entry, font = ('Arial black', 22), fg = 'light blue')

update\_button.grid(row = 8, column = 4)

# Entry widget for input

user\_entry = Entry(main\_screen, borderwidth = 5)

user\_entry.grid(row = 1, column = 1)

email\_entry = Entry(main\_screen, borderwidth = 5)

email\_entry.grid(row = 1, column = 3)

#set up a radiobutton question

selection = StringVar()

Radiobutton(option\_frame, text = 'quotes', variable = selection, value = 'quotes' ).pack(anchor = W)

Radiobutton(option\_frame, text = 'comma', variable = selection, value = 'comma').pack(anchor = W)

Radiobutton(option\_frame, text = 'colon', variable = selection, value = 'colon').pack(anchor = W)

Radiobutton(option\_frame, text = 'asterisk', variable = selection, value = 'asterisk').pack(anchor = W)

radio\_button = Button(option\_frame, text = 'Check answer', bg = 'red', command = lambda:check(selection.get()))

radio\_button.pack(anchor = W)

#slider widget is called Scale

#define vertical slider for year entry

tall = Scale(main\_screen, from\_ = 2000, to = 2030)

tall.grid(row = 4, column = 3)

year\_label = Label(main\_screen, textvariable = year\_num)

year\_label.grid(row = 6, column = 3)

year\_button = Button (main\_screen, text = 'Submit Year', bg = 'red', command = get\_year)

year\_button.grid(row = 5, column = 3)

#create a drop down menu for lunch extras

#choose from pizza, hambuger, or salad

lunch\_options = OptionMenu(main\_screen, entree\_clicked, 'Pizza', 'Hamburger', 'Salad')

lunch\_options.grid(row = 7, column = 0)

#create drop down menu for students entered

student\_menu = OptionMenu(main\_screen, student\_clicked,\*student\_options)

student\_menu.grid(row = 7, column = 3, pady= 5)

#scale widget for ratings entry

wide = Scale(main\_screen, from\_ = 0, to = 100, orient = HORIZONTAL)

wide.grid(row = 4, column = 2)

rate\_label = Label(main\_screen, textvariable = rate\_num)

rate\_label.grid(row = 6, column = 2)

rating = Button(main\_screen, text = "Submit Rating", bg = 'red', command = get\_rating)

rating.grid(row = 5, column = 2)

#create check boxes for a pizza order

#default variable is 1 or 0, set variariable with attributes onvalue or offvalue

check\_box1 = Checkbutton(main\_screen, text = 'Cheese', variable = has\_cheese)

check\_box1.grid(row = 7, column = 1)

check\_box2 = Checkbutton(main\_screen, text = 'Peppers', variable = has\_peppers)

check\_box2.grid(row = 8, column = 1)

check\_box3 = Checkbutton(main\_screen, text = 'Mushrooms', variable = has\_mushrooms)

check\_box3.grid(row = 9, column = 1)

check\_box4 = Checkbutton(main\_screen, text = 'Tomatoes', variable = has\_tomatoes)

check\_box4.grid(row = 10, column = 1)

lunch\_order = Button(main\_screen, text = 'Order Lunch', command = get\_lunch, pady = 5, bg = 'red')

lunch\_order.grid(row = 11, column = 1, pady = 10)

query\_students = Button(main\_screen, text = 'Show entries', command = show\_entries)

query\_students.grid(row = 14, column = 0)

main\_screen.mainloop()

# Fourth Project – Exercises

## Change the scale of the wide slider to range from one to five.

## Change the scale of the tall slider to range from 2003 to 2029

## Change the text of the message box to read ‘great job’ when correct and ‘try again’

## Add sound effects for correct and incorrect answer to the question label frame

## Add another topping to the listand update all the corresponding features such as the order in the get\_lunch() function.

# Fourth Assessment

In the live class, an assessment will be done during the QUESTION-AND-ANSWER period. For self-paced learners, the assessment questions are at the end of the eBook.

## [JUmp to The fourth assessment](#_Fourth_assessment)

# Collaboration Activity

## Add project files to github

## Commit changes to your repository.

## Push the changes to your repo from your local drive

## Talk to someone about your project

# Fourth Project – Deployment Plan

## Use pyinstaller to make onefile version of the project

## Zip the files into a folder

## Use NSIS to create your executable program

## Share your project!

# Enhance and Make it Yours!

## Think of how you can make the widgets app better.

## Is there a feature that you can put on your app with the widgets you learned?

## Brand your idea for the widgets to add to your portfolio.

# First assessment

Please answer the following questions about the **Basic Calculator** project.

|  |  |
| --- | --- |
| 1. Which keyword can be used to access variables and functions of another python file? | |
| A | input |
| B | import |
| C | as |
| D | def |

|  |  |
| --- | --- |
| 2. Which keyword is used for making an alias module name? | |
| A | as |
| B | for |
| C | import |
| D | from |

|  |  |
| --- | --- |
| 3. Which symbol is used to assign a value to a variable | |
| A | [] |
| B | : |
| C | “” |
| D | = |

|  |  |
| --- | --- |
| 4. Which option is the correct way to access the “move” function within a module called foo? | |
| A | move=>foo |
| B | move(foo) |
| C | foo.move() |
| D | foo(move) |

|  |  |
| --- | --- |
| 5. What symbol is used at the end of a loop declaration? | |
| A | colon |
| B | hyphen |
| C | arrow |
| D | parenthesis |

|  |  |
| --- | --- |
| 6. What keywords are used to implement exception handling in python? | |
| A | try/catch |
| B | try/except |
| C | try/fail |
| D | try/error |

|  |  |
| --- | --- |
| 7. Which statements require a colon in their declaration? | |
| A | variable, loop, conditional, dictionary, function |
| B | list, loop, conditional, dictionary, function |
| C | loop, conditional, dictionary, function |
| D | array, loop, conditional, function |

|  |  |
| --- | --- |
| 8. Which option shows the correct loop statement signature line? | |
| A | for item as items: |
| B | while item in items: |
| C | while item as items: |
| D | for item in items: |

|  |  |
| --- | --- |
| 9. What will be the outcome of the following assignment statement?  button7 = Button(root, text = "7", command=lambda: click\_action(7),padx=40,pady=20, bg='navy', fg='white') | |
| A | A button displayed for button7 with a navy background and white text. |
| B | A button displayed for button7 with a navy background and white border. |
| C | A button assigned to button7 with a lambda function called click\_action and the number 7 as the argument. |
| D | A button assigned to button7 with a lambda function called click\_action and “7” as the argument. |

|  |  |
| --- | --- |
| 10. What keyword is used to allow access to a variable outside of a function definition? | |
| A | local |
| B | global |
| C | entry |
| D | access |

# Second assessment

Please answer the following python questions about the **Prime Time Game** project.

|  |  |
| --- | --- |
| 1. What module can be used to pick a sample number within a range of numbers? | |
| A | number |
| B | list |
| C | range |
| D | random |

|  |  |
| --- | --- |
| 2. What type of statement will execute a block of code when a condition is True? | |
| A | if |
| B | when |
| C | from |
| D | for |

|  |  |
| --- | --- |
| 3. Which module allows python to display image files? | |
| A | display |
| B | ImageTk |
| C | PIL |
| D | photo |

|  |  |
| --- | --- |
| 4. Which python symbol is used to compare two values in an **if** statement? | |
| A | = |
| B | == |
| C | === |
| D | ? |

|  |  |
| --- | --- |
| 5. Which option is a function call used to find the length of a list called foo and assigns it to u? | |
| A | u=length(foo) |
| B | u=>size(foo) |
| C | u=>typeof(foo) |
| D | u=len(foo) |

|  |  |
| --- | --- |
| 6. What does a Unicode character start with? | |
| A | u |
| B | \u |
| C | “” |
| D | ‘/’ |

|  |  |
| --- | --- |
| 7. Which module allows python to add sound to an application? | |
| A | sound |
| B | play |
| C | playsound |
| D | music |

|  |  |
| --- | --- |
| 8. Which statement is an incorrect function definition called move? | |
| A | define move: |
| B | define move() |
| C | def move(): |
| D | move(num): |

|  |  |
| --- | --- |
| 9. What will be the outcome of the following code when blink is an odd number?  if blink%2==0:     showScore.config(fg='red')     statusMsg.config(fg = 'red')  else:  showScore.config(fg='white')  statusMsg.config(fg ='white') | |
| A | The text of showScore will be white and the text of statusMsg will be white. |
| B | The text of showScore will be red and the text of statusMsg will be red. |
| C | There will be a syntax error. |
| D | Outcome cannot be determined. |

|  |  |
| --- | --- |
| 10. What is the boolean literal value if a condition has been met? | |
| A | bool |
| B | true |
| C | True |
| D | False |

# Third assessment

Please answer the following questions about the Photo Gallery project.

|  |  |
| --- | --- |
| 1. Which option is the correct list declaration for the variable foo\_list? | |
| A | foo\_list = [] |
| B | foo=list() |
| C | foo: “list” |
| D | foo\_list => {} |

|  |  |
| --- | --- |
| 2. How many lines of code can be included in a **for** loop? | |
| A | 4 |
| B | unlimited |
| C | 1 |
| D | 100 |

|  |  |
| --- | --- |
| 3. Which python function call will give access to a file called foo.txt? | |
| A | file = open(foo.txt) |
| B | file = open(‘foo.txt’) |
| C | file = access(foo.txt) |
| D | file = access(‘foo.txt’) |

|  |  |
| --- | --- |
| 4. What is the outcome, for the following function call, of a list called foo\_list?  foo\_list.append(‘bar’) | |
| A | A bar will be inserted at the front of foo\_list. |
| B | A bar will be inserted after the first element in foo\_list. |
| C | A syntax error will occur. |
| D | A string, ‘bar’, will be added to the end of foo\_list. |

|  |  |
| --- | --- |
| 5. Which function call properly passes in an integer argument called **num?** | |
| A | num=foo(‘bar’) |
| B | num= lambda: foo() |
| C | foo(num) |
| D | foo({‘num’: ‘bar’}) |

|  |  |
| --- | --- |
| 6. What is the keyword for sending results of a function call? | |
| A | return |
| B | pass |
| C | result |
| D | send |

|  |  |
| --- | --- |
| 7. What is the attribute name for adding a photo to a label? | |
| A | photo |
| B | image |
| C | open |
| D | ImageTk |

|  |  |
| --- | --- |
| 8. Which function call is used to end the access of a file variable called foo? | |
| A | end.foo() |
| B | foo.end() |
| C | foo.close() |
| D | close(foo) |

|  |  |
| --- | --- |
| 9. Which other keyword goes on the same line with a loop besides **for**? | |
| A | in |
| B | with |
| C | as |
| D | while |

|  |  |
| --- | --- |
| 10. Which function call can be used to modify the text of a label widget, bar\_label, to ‘foo’? | |
| A | bar\_label.change(text = ‘foo’ ) |
| B | bar\_lable.modify(text = ‘foo’) |
| C | bar\_label.update(text = ‘foo’) |
| D | bar\_label.config(text = ‘foo’) |

# Fourth assessment

Please answer the following questions about the Widgets project.

|  |  |
| --- | --- |
| 1. Which function in sqlite3 is used to render queries? | |
| A | execute |
| B | run |
| C | loop |
| D | fetchall |

|  |  |
| --- | --- |
| 2. Which function in sqlite3 will list the output of a query? | |
| A | execute |
| B | fetchall |
| C | cursor |
| D | connect |

|  |  |
| --- | --- |
| 3. Which collection of objects uses key/value pair(s) for its declaration? | |
| A | list |
| B | tuple |
| C | dictionary |
| D | array |

|  |  |
| --- | --- |
| 4. What type of widget allows you to see a list come down when you click it. | |
| A | DropMenu |
| B | OptionMenu |
| C | MenuOption |
| D | ListOption |

|  |  |
| --- | --- |
| 5. Which type of layout put widgets in place by automatic order. | |
| A | place |
| B | pack |
| C | push |
| D | grid |

|  |  |
| --- | --- |
| 6. Which widget allows you to select options within a range of values? | |
| A | Label |
| B | Button |
| C | Scale |
| D | RadioButton |

|  |  |
| --- | --- |
| 7. Which function is used to open a database? | |
| A | open |
| B | connect |
| C | execute |
| D | cursor |

|  |  |
| --- | --- |
| 8. Which function call is used to end the use of a database variable called foo? | |
| A | end(foo.db) |
| B | foo.close() |
| C | pass(‘foo.db’) |
| D | open(foo.db) |

|  |  |
| --- | --- |
| 9. What is the outcome of the following python script?  sql\_db = sqlite3.connect('foo.db')  control = sql\_db.cursor()  control.execute('SELECT oid,\* FROM users')  entries = control.fetchall()  for entry in entries:      user\_options.append(entry)    sql\_db.commit()  sql\_db.close() | |
| A | A list of entries is added to the users table from the foo database. |
| B | A list of entries is fetched from the foo table in the users database. |
| C | The user\_options list is populated with entries from the users table in the foo database. |
| D | The user\_options list is selected as the entry point for the foo table in the users database. |

|  |  |
| --- | --- |
| 10. Which widget allows you to display a pop-up window with an **ok** button and informative text? | |
| A | messagebox |
| B | PopUpBox |
| C | dialogBox |
| D | MessageWindow |

|  |  |
| --- | --- |
| 11. What do you do after opening a database, but before executing a query? | |
| A | Set up a cursor |
| B | Take a lunch break |
| C | Insert a comment |
| D | Fetch all the entries |

|  |  |
| --- | --- |
| 12. Which statements can be used to query a database in sqlite3? | |
| A | INSERT INTO, UPDATE, OPEN, DELETE |
| B | INSERT INTO, UPDATE, LIST, DELETE |
| C | INSERT INTO, SELECT, UPDATE, DELETE |
| D | INSERT INTO, FETCHALL, UPDATE, DELETE |

|  |  |
| --- | --- |
| 13. Which sqlite3 keyword is used to change the value of a field in a record of a database table? | |
| A | SET |
| B | FROM |
| C | SELECT |
| D | DELETE |

|  |  |
| --- | --- |
| 14. Which software turns your zip file into an executable file for deployment? | |
| A | NSIS |
| B | pyinstaller |
| C | PIP |
| D | PIL |

|  |  |
| --- | --- |
| 15. Which module allows you to prepare your python application for deployment. | |
| A | pyinstaller |
| B | pygame |
| C | PIL |
| D | deploy |

# Summary

You have learned a lot about Python project development if you have followed the code reviews and did the exercises and assessments. You have four projects in your portfolio that you can list on your resume or link to your GitHub repository. The basic fundamentals of programming have been covered. Good job. Now go and be a spectacular developer!

## variables

## flow control

## Operators

## Loops and iterables

## Data structures

## collections

## Functions

## basic File management

## Basic database management

## desktop application deployment