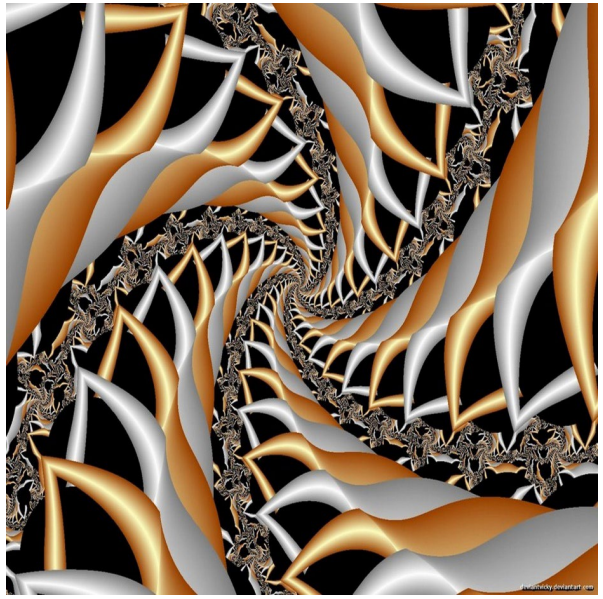
COSC 1437
Programming Fundamentals II - Python

Examination 2
Programming Problem Specs

Tic – Tac – Toe
with Tkinter



COSC 1437,

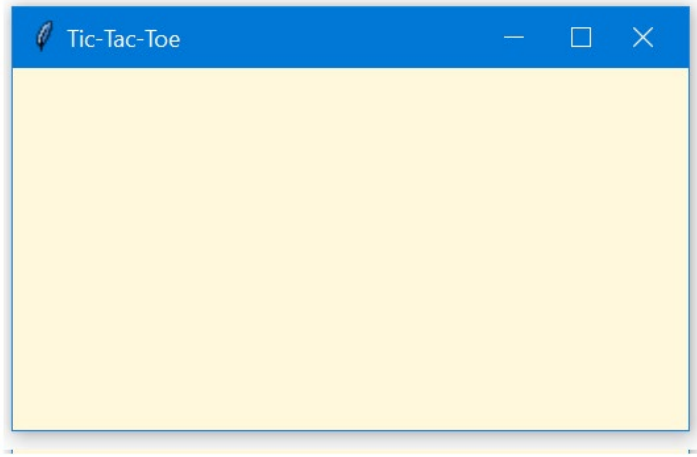
Exam 2
Tic – Tac – Toe with Tkinter

1. Create your own class name
2. Write a docstring for the class
3. Create a simple 3 – item menu (it doesn't have to have functionality).
   Specify in the comments what the 3 menu items are
   You will see where it is used later in the code.
4. For all functions write your docstring (I have not done this for you in this guide.).

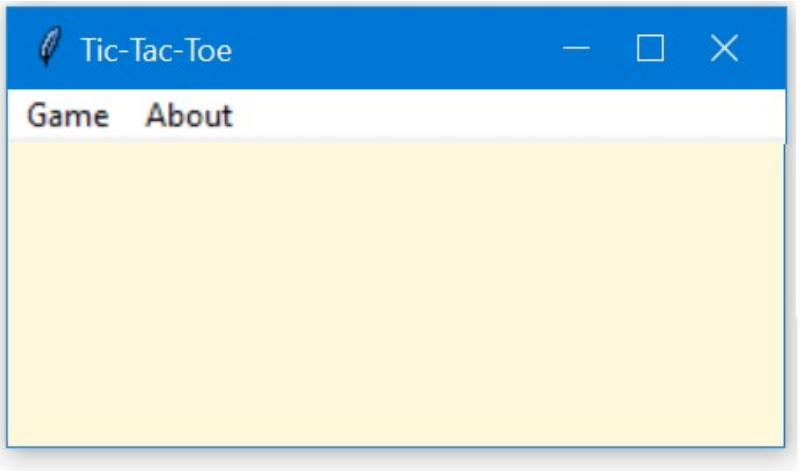
Comment:
After you get the buttons placed, depending on your computer screen size, resolution, and OS, you may need to change the default size of the window.

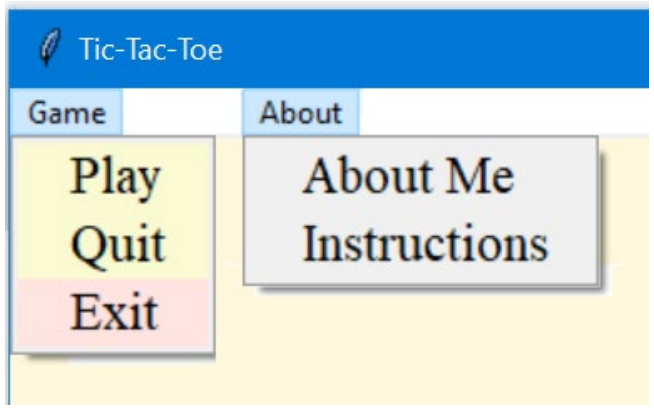
Step 1:

```
from tkinter import *

class __________:

    def aboutMe(self):
        pass

    def __init__(self):
        self.window = Tk()
        self.window.title("Tic-Tac-Toe")
        self.window.geometry("1200x1200")

        self.window.option_add("*Font", 'Times 16')
        self.window.configure(bg='#FFF8DC')
        self.window.resizable(False, False)

        self.moves = 0

        self.window.mainloop()

TicTacToe() # Create GUI
```



Exam 2: Tkinter Tic – Tac - Toe





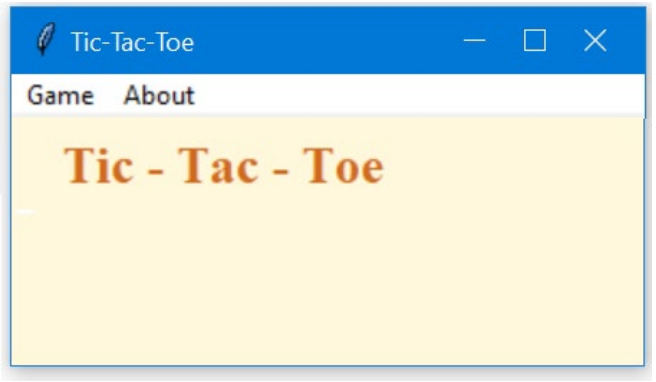
Step 2: Add Menu Options
See **Appendix 1** and **Appendix 2**

Test the Exit Menu Option

Step 3: Download Image Files, Create folder named **images**

```
# Create images, Use relative file addressing. Create a folder images on your computer.
self.ImageX    = PhotoImage(file = r"images\PythonX.PNG")
self.ImageO    = PhotoImage(file = r"images\PythonO.PNG")
self.ImageBase = PhotoImage(file = r"images\PythonBase.PNG")
self.ImageH    = PhotoImage(file = r"images\PythonHorizontal2.PNG")
self.ImageV    = PhotoImage(file = r"images\PythonVertical.PNG")
# You must use these filenames and folder name.
```

Exam 2: Tkinter Tic – Tac - Toe



Step 4: Add Label

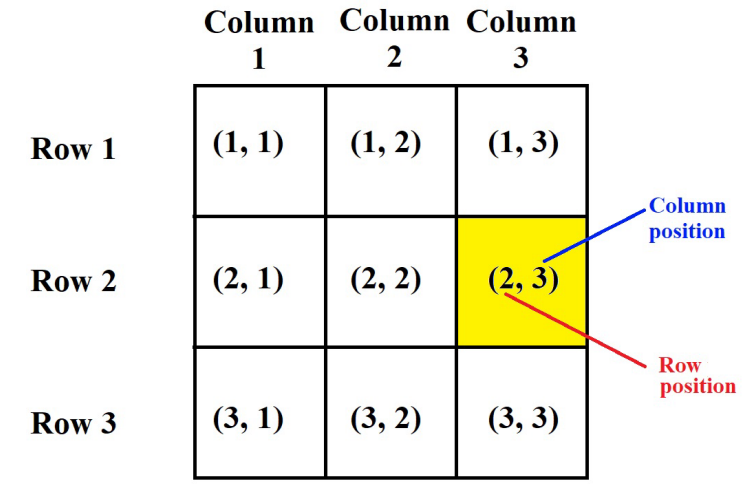
```
# create your header label (choose your own variable name)
________ = Label(self.window,
    text  = 'Tic - Tac - Toe',font  = ('Times 18 bold'),
    bg    = '#FFF8DC',fg    = '#D2691E’)

________.grid(row = 0, column = 1, padx = (20,0),
    pady = (5,0), sticky = 'WS', columnspan = 7)
```

Step 5: Set Default Attributes

```
# set default values
self.b11Val, self.b12Val, self.b13Val = 0, 0, 0
self.b21Val, self.b22Val, self.b23Val = 0, 0, 0
self.b31Val, self.b32Val, self.b33Val = 0, 0, 0
```

Comment:

We are using matrix notation for the 3 x 3 Tic-Tac-To grid



In our code specs, we are refer to this square as b23 (i.e., second row and third column).
The same schema holds for references to the other cells/square of the grid.

Exam 2: Tkinter Tic – Tac - Toe

Step 6: Adding to the `__init__` method: Add first Button Image

```
# build and place default buttons
self.b11 = Button(self.window, image = self.ImageBase,
   command = self.cycle11, highlightthickness = 0, bd = 0)
self.b11.grid(row = 1, column = 1, sticky = 'W',ipadx=0,
   ipady=0, padx=(20,0), pady=(50,0))
```

Comment:

Notice our pattern here, we will need it for the other 8 buttons. You can really use any schema you want to remember which (command) functions need to be associated with which attributes (here the cell/square).

```
self.b11 = Button(self.window, image = self.ImageBase,
   command = self.cycle11, highlightthickness = 0, bd = 0)
self.b11.grid(row = 1, column = 1, sticky = 'W',ipadx=0,
   ipady=0, padx=(20,0), pady=(50,0))
```

Cell (1,1) which, for us, we see in the code appearing in b11 and cycle11 and ... row = 1, column = 1

Hence, for Cell (2,3) for us -- sticking to the same schema -- would see in the code appearing in b23 and cycle23 and ... row = 2, column = 3

Exam 2: Tkinter Tic – Tac - Toe

Step 7: Adding to the `__init__` method: Add Radio Buttons

```
self.radioVar = IntVar()                                    # Add descriptive comment
# Add descriptive comment
self.radio1 = Radiobutton(self.window, font = ('Times' , 25),text="X",
    variable=self.radioVar, value=1, command = self.select)

# Add descriptive comment
self.radio2 = Radiobutton(self.window, font = ('Times' , 25),text="O",
    variable=self.radioVar, value=0, command = self.select)
# Add descriptive comment
self.radio1.grid(row = 1,  column = 6, sticky = 'N',pady=(50,0))
self.radio2.grid(row = 1,  column = 7, sticky = 'N',pady=(50,0))
```

Comment:

You need to add your comment for `# Add descriptive comment` which demonstrates your understanding of what this line of code does.

Exam 2: Tkinter Tic – Tac - Toe

Step 8: Write `cycle11` method

```
def cycle11(self):
    self.moves += 1                                          # Add descriptive comment
    if self.radioVar.get() == 1:
         self.b11.configure(image = self.ImageX )
         self.b11Val = 1                                     # Add descriptive comment
         play = "X"
    else:
         self.b11.configure(image = self.ImageO )
         self.b11Val = -1                                     # Add descriptive comment
         play = "O"
    self.b11["state"] = "disabled"
    self.report(f"{play} played a move at Square (1,1)") # Add descriptive comment
```

Comment: Your `report` function will call the function to check if there is a winner (See Step 9)

You need to add your comment for `# Add descriptive comment` which demonstrates your understanding of what this line of code does.

Step 9: Write a method to determine if there's a winner

Exam 2: Tkinter Tic – Tac - Toe

```
def _________(self):
    hch1 = self.b11Val + self.b12Val + self.b13Val                    # Add descriptive comment
    hch2 = self.b21Val + self.b22Val + self.b23Val
    hch3 = self.b31Val + self.b32Val + self.b33Val
    vch1 = self.b11Val + self.b21Val + self.b31Val
    vch2 = self.b12Val + self.b22Val + self.b32Val
    vch3 = self.b13Val + self.b23Val + self.b33Val
    dch1 = self.b11Val + self.b22Val + self.b33Val
    dch2 = self.b13Val + self.b22Val + self.b31Val
    # Add descriptive comment
    winnerX = hch1== 3 or hch2== 3 or hch3== 3 or vch1== 3 or vch2== 3 or vch3== 3 or dch1== 3 or dch2== 3
    winnerO = hch1==-3 or hch2==-3 or hch3==-3 or vch1==-3 or vch2==-3 or vch3==-3 or dch1==-3 or dch2==-3
    if winnerX:
        self.winner = "X"
        self.disableBoard()
        return True
    elif winnerO:
        self.winner = "O"
        self.disableBoard()
        return True
    else:
        if self.moves == 9:
           print("There is a tie.")
           # write code to add Label to the GUI window that there is a tie
        else:
           return False
```

Exam 2: Tkinter Tic – Tac - Toe

Step 10: Write your reporting function

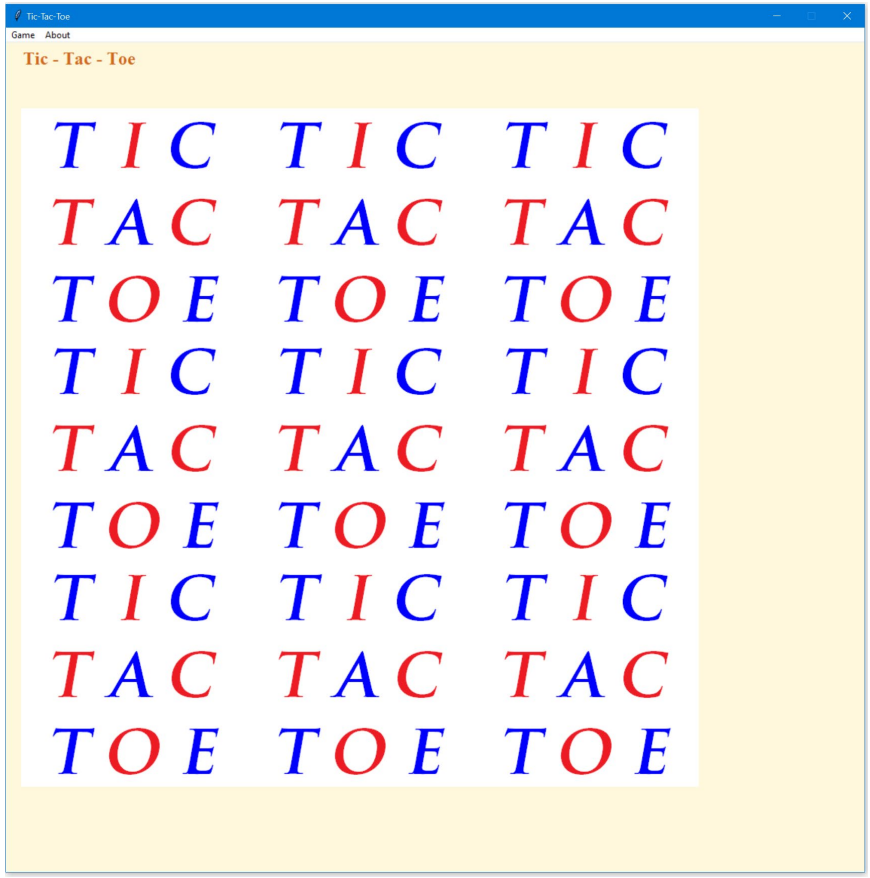
```
def report(self, moveInfo):
        print(moveInfo)
        # Add descriptive comment
        if self._________():
            print(f"{self.winner} has won the game")    #Output to Console
        else:
            print("No winner yet")
```

Comment: This is a call to the method you wrote in Step 9.

You need to add your comment for `# Add descriptive comment` which demonstrates your understanding of what this line of code does.

Exam 2: Tkinter Tic – Tac - Toe

Step 11: Add the other 8 buttons
See **Appendix 3**



Step 12: Using **`cycle11`** as a template, write the other button command methods

```
self.cycle12
self.cycle13
self.cycle21
self.cycle22
self.cycle23
self.cycle31
self.cycle32
self.cycle33
```

Step 13: Write **`disableBoard`** method

```
def disableBoard(self):
      self.b11["state"] = "disabled"
      self.b12["state"] = "disabled"
      self.b13["state"] = "disabled"
      self.b21["state"] = "disabled"
      self.b22["state"] = "disabled"
      self.b23["state"] = "disabled"
      self.b31["state"] = "disabled"
      self.b32["state"] = "disabled"
      self.b33["state"] = "disabled"
```

Exam 2: Tkinter Tic – Tac - Toe

Step 14: Adding to the __init__ method: Add Hashtag Grid

```
vbar1 = Label(self.window, image = self.ImageV, highlightthickness = 0, bd = 0 )
vbar1.grid(row = 1, column = 2, sticky = 'W', rowspan = 6, ipadx=0, ipady=0, pady=(50,0))

vbar2 = Label(self.window, image = self.ImageV , highlightthickness = 0, bd = 0)
vbar2.grid(row = 1, column = 4, sticky = 'W', rowspan = 6, ipadx=0, ipady=0, pady=(50,0))

hbar1 = Label(self.window, image = self.ImageH, highlightthickness = 0, bd = 0 )
hbar1.grid(row = 2, column = 0, sticky = 'W', columnspan = 6, ipadx=0, ipady=0,padx=(20,0))

hbar2 = Label(self.window, image = self.ImageH , highlightthickness = 0, bd = 0)
hbar2.grid(row = 4, column = 0, sticky = 'W', columnspan = 6, ipadx=0, ipady=0, padx=(20,0))
```

Appendix 1: Adding to the __init__ method

Exam 2: Tkinter Tic – Tac - Toe

```
# Create a menu bar
self.menubar = Menu(self.window) # Create a Mmenu
self.window.config(menu = self.menubar) # Display the menu bar

self.GameMenu = Menu(self.menubar, tearoff="off")
self.menubar.add_cascade(label="Game", menu=self.GameMenu)

self.AboutMenu = Menu(self.menubar, tearoff="off")
self.menubar.add_cascade(label="About", menu=self.AboutMenu)
```

Appendix 2: Adding to the __init__ method

```
# Add menu items to Game
self.GameMenu.add_command(label = "Play", background ='#FAFAD2', activebackground = '#FFFFFF',
activeforeground = '#FF0000')
self.GameMenu.add_command(label = "Quit", background ='#FAFAD2')
self.GameMenu.add_command(label = "Exit", background ='#FFE4E1', command = exit)

# Add menu items to About
self.AboutMenu.add_command(label = "________", command = self.aboutMe)
# add two more menu items, per the instructions given
self.window.config(menu=self.menubar)
```

Appendix 3: Adding to the __init__ method

```
# build and place default buttons
self.b11 = Button(self.window, image = self.ImageBase, command = self.cycle11, highlightthickness = 0, bd = 0)
self.b11.grid(row = 1, column = 1, sticky = 'W', ipadx=0, ipady=0, padx=(20,0), pady=(50,0))

self.b12 = Button(self.window, image = self.ImageBase, command = self.cycle12, highlightthickness = 0, bd = 0)
self.b12.grid(row = 1, column = 3, sticky = 'W', ipadx=0, ipady=0, pady=(50,0))

self.b13 = Button(self.window, image = self.ImageBase, command = self.cycle13, highlightthickness = 0, bd = 0)
self.b13.grid(row = 1, column = 5, sticky = 'W', ipadx=0, ipady=0, pady=(50,0))

self.b21 = Button(self.window, image = self.ImageBase, command = self.cycle21, highlightthickness = 0, bd = 0)
self.b21.grid(row = 3, column = 1, sticky = 'W', ipadx=0, ipady=0, padx=(20,0))

self.b22 = Button(self.window, image = self.ImageBase, command = self.cycle22, highlightthickness = 0, bd = 0)
self.b22.grid(row = 3, column = 3, sticky = 'W', ipadx=0, ipady=0)

self.b23 = Button(self.window, image = self.ImageBase, command = self.cycle23, highlightthickness = 0, bd = 0)
self.b23.grid(row = 3, column = 5, sticky = 'W', ipadx=0, ipady=0)

self.b31 = Button(self.window, image = self.ImageBase, command = self.cycle31, highlightthickness = 0, bd = 0)
self.b31.grid(row = 5, column = 1, sticky = 'W', ipadx=0, ipady=0, padx=(20,0))

self.b32 = Button(self.window, image = self.ImageBase, command = self.cycle32, highlightthickness = 0, bd = 0)
self.b32.grid(row = 5, column = 3, sticky = 'W', ipadx=0, ipady=0)

self.b33 = Button(self.window, image = self.ImageBase, command = self.cycle33, highlightthickness = 0, bd = 0)
self.b33.grid(row = 5, column = 5, sticky = 'W', ipadx=0, ipady=0)
```