Day - 28

**Tkinter: Dynamic Typing &**

**the Pomodoro GUI App**

Canvas, Add Images to Tkinter , Dynamic Typing, Color Hunt

**28.1 Python is dynamically typed**

-Python is strongly typed; it holds on to the data type of the variable. when you assign a value, it knows that it's a ***string***, ***int*** or ***float***. if you are trying to do something that's not meant for a type of value, it's gonna print ***'type error'***.

-Python is dynamically typed; it allows you to change the data type of variable, just by assigning it to different type of value(changing the content). Other languange doesn't allow it. (maintenance, test, reuse, planning time)

-Opposite is Static typed programming languages. (fast, strict, team)

a =3 -> int

a= "Hi" ->changed to string

Dynamic typing allows the code below works, count\_sec was int and it works inside string.

if count\_sec < 10:

count\_sec = f"0{count\_sec}"

**28.2 Project: Pomodoro GUI – step 1**

**Setup the window and Background image**

**import**  tkinter

#*---------------------------- CONSTANTS ------------------------------- #*

PINK = "#e2979c"

RED = "#e7305b"

GREEN = "#9bdeac"

YELLOW = "#f7f5dd"

FONT\_NAME = "Courier"

WORK\_MIN = 25

SHORT\_BREAK\_MIN = 5

LONG\_BREAK\_MIN = 20

#*---------------------------- TIMER RESET ------------------------------- #*

#*---------------------------- TIMER MECHANISM ------------------------------- #*

#*---------------------------- COUNTDOWN MECHANISM ------------------------------- #*

#*---------------------------- UI SETUP ------------------------------- #*

winDow = tkinter**.Tk**()

winDow**.minsize**(height = 500, width = 600)

winDow**.title**("Pomdoro GUI app")

#*Back ground color*

winDow**.config**(bg = YELLOW)

winDow**.config**(padx =100, pady = 50)

#*including image*

    #*PhotoImage is needed to attach image to tkinter*

pic\_tomato = tkinter**.PhotoImage**(file= "./tomato.png")

    #*crfreating Canvas for image: Its like a artist canvas, it allows layering*

can\_vas = tkinter**.Canvas**(width = 220, height = 250, bg = YELLOW, highlightthickness= 0)

            #*highlightthickness= 0 to disable canvas border*

    #*attaching image to canvas. "bg" for background color, "image" to insert image*

    #*must include x, y position*

can\_vas**.create\_image**(110, 125, image=pic\_tomato)

    #*wont show unless use a geometry manager: pack(), place() or grid()*

#*can\_vas.grid(column = 0, row = 0)*

can\_vas**.pack**()

#*Adding text top of image. Modify the text, use "fill" for color*

can\_vas**.create\_text**(110, 135, text = "00:00", fill = "white", font = (FONT\_NAME, 20, "bold"))

winDow**.mainloop**()

#*python pomodoro\_main.py*

**28.3 Challenge - Complete the Application's User Interface (UI)**

#*buttons and griding*

#*fg - foreground color/ text color , bg back-ground color*

heading = tkinter**.Label**(text = "Timer", bg = YELLOW, fg =  GREEN, font = (FONT\_NAME, 36, "bold") )

heading**.grid**(column = 1, row = 0)

#*# "bd" or "borderwidth" for border thickness*

button\_start = tkinter**.Button**(text = "Start", bg = RED, fg =  "white",  borderwidth = 0, font = (FONT\_NAME, 12, "bold") )

button\_start**.grid**(column = 0, row = 2)

button\_reset = tkinter**.Button**(text = "Reset", bg = RED, fg =  "white", bd = 0, font = (FONT\_NAME, 12, "bold") )

button\_reset**.grid**(column = 2, row = 2)

check = tkinter**.Label**(text = "✔", bg = YELLOW, fg = GREEN ,font = (FONT\_NAME, 12))

check**.grid**(column = 1, row = 3)

**28.4 Add a Count Down Mechanism**

Cannot use ***for loop*** in ***Tkinter GUI*** wher ***mainloop()*** is used, instead

* Schedule an Action with Tkinter ***after()*** method
* All Tkinter widgets have the ***after()*** method with the following syntax:

**after**(delay, callback=None)

* The ***after()*** method calls the ***callback*** function once after a ***delay*** milliseconds (ms) within Tkinter’s main loop.
* If you don’t provide the ***callback***, the ***after()*** method behaves like the ***time.sleep()*** function. However, the ***after()*** method uses the ***millisecond*** instead of the ***second*** as the unit.
* However, we could use ***time.sleep()*** with a loop, that doesnot work with ***GUI***, because ***mainloop()*** is alredy running loop. The reason is that the ***sleep()*** function suspends the main thread execution. Therefore, ***Tkinter*** couldn’t update the ***GUI***.
* Event driven GUI:
* Event driven: GUI keeps checking on the events and when it detects the ***event(click etc)***, it's ready to react.'***window.mainloop()***' enables it. It loops through every millisecond to check a new event.
* if we use another ***'while*** ***loop'*** inside '***window.mainloop***', it won't be able to reach the ***mainloop*** and it won't even launch the app.

**import** math

#*---------------------------- TIMER RESET ------------------------------- #*

#*---------------------------- TIMER MECHANISM ------------------------------- #*

    #*To invoke count\_down() when button is clicked*

**def** **start\_timer**():

**count\_down**(60 \* WORK\_MIN)

#*---------------------------- COUNTDOWN MECHANISM ------------------------------- #*

**def** **count\_down**(count):

**if** count **>=** 0:

        remain\_min = math**.floor**(count/60)

        remain\_sec = count % 60

        #*can\_vas.itemconfig() is like Label.coonfig()*

        can\_vas**.itemconfig**(timer\_text, text = f"{remain\_min}:{remain\_sec}")

        winDow**.after**(1000, count\_down, count -1 )

#*---------------------------- UI SETUP ------------------------------- #*

winDow = tkinter**.Tk**()

winDow**.minsize**(height = 300, width = 420)

winDow**.title**("Pomdoro GUI app")

#*Back ground color*

winDow**.config**(bg = YELLOW)

winDow**.config**(padx =100, pady = 50)

#*including image*

    #*PhotoImage is needed to attach image to tkinter*

pic\_tomato = tkinter**.PhotoImage**(file= "./tomato.png")

    #*crfreating Canvas for image: Its like a artist canvas, it allows layering*

can\_vas = tkinter**.Canvas**(width = 220, height = 250, bg = YELLOW, highlightthickness= 0)

            #*highlightthickness= 0 to disable canvas border*

    #*attaching image to canvas. "bg" for background color, "image" to insert image*

    #*must include x, y position*

can\_vas**.create\_image**(110, 125, image=pic\_tomato)

    #*wont show unless use a geometry manager: pack(), place() or grid()*

#*Grid is relative. To center the position  padding added to window*

can\_vas**.grid**(column = 1, row = 1)

#*Adding text top of image. Modify the text, use "fill" for color*

timer\_text = can\_vas**.create\_text**(110, 135, text = "00:00", fill = "white", font = (FONT\_NAME, 20, "bold"))

# timer\_text is uset to change the text from the count\_down()

#*buttons and griding*

#*fg - foreground color/ text color , bg back-ground color*

heading = tkinter**.Label**(text = "Timer", bg = YELLOW, fg =  GREEN, font = (FONT\_NAME, 36, "bold") )

heading**.grid**(column = 1, row = 0)

#*# "bd" or "borderwidth" for border thickness*

button\_start = tkinter**.Button**(text = "Start", bg = RED, fg =  "white",  borderwidth = 0, font = (FONT\_NAME, 12, "bold") )

    #*start\_timer will invoke count\_down() when button is clicked*

button\_start**.config**(command = start\_timer)

button\_start**.grid**(column = 0, row = 2)

button\_reset = tkinter**.Button**(text = "Reset", bg = RED, fg =  "white", bd = 0, font = (FONT\_NAME, 12, "bold") )

button\_reset**.grid**(column = 2, row = 2)

check = tkinter**.Label**(text = "✔", bg = YELLOW, fg = GREEN ,font = (FONT\_NAME, 12))

check**.grid**(column = 1, row = 3)

winDow**.mainloop**()

**28.5 Dynamic Typing Explained**

One hand Python is strongly typed: following does not work, Python can detect the type

Str = 2 + 'e'

On the other hand Python is also a dynamic typed language. The data type of the variable A just changes during the assign:

A= 34

A = "name"

For this reason following works:

**def** **count\_down**(count):

**if** count **>=** 0:

        remain\_min = math**.floor**(count/60)

        remain\_sec = count % 60

**if remain\_sec <= 9:**

**#*remain\_sec = "00" # dynamic type occuring "int" to "string" auto-conversion***

**#*remain\_sec = "0" + str(remain\_sec) # dynamic type occuring "int" to "string" auto-conversion***

**remain\_sec = f"0{remain\_sec}"   #*dynamic type occuring "int" to "string" auto-conversion***

        #*can\_vas.itemconfig() is like Label.coonfig()*

        can\_vas**.itemconfig**(timer\_text, text = f"{remain\_min}:{remain\_sec}")

        winDow**.after**(100, count\_down, count -1 )

**28.6 Setting Different Timer Sessions and Values**

#*---------------------------- TIMER MECHANISM ------------------------------- #*

reps = 0

    #*To invoke count\_down() when button is clicked*

**def** **start\_timer**():

**global** reps

    reps += 1

    work\_time = WORK\_MIN \* 60

    short\_break = SHORT\_BREAK\_MIN \* 60

    long\_break = LONG\_BREAK\_MIN \* 60

**if** (reps **>** 0) **and** (reps % 8 **==** 0):

**count\_down**(long\_break)

        heading**.config**(text = "LONG Break", fg =  RED)

        reps = 0

**elif** (reps **>** 0) **and** (reps % 2 **==** 0):

**count\_down**(short\_break)

        heading**.config**(text = "SHORT Break", fg =  PINK)

**else**:

**count\_down**(work\_time)

        heading**.config**(text = "WORK time", fg =  GREEN)

**print**(reps)

#*---------------------------- COUNTDOWN MECHANISM ------------------------------- #*

**def** **count\_down**(count):

**if** count **>** 0:

        remain\_min = math**.floor**(count/60)

        remain\_sec = count % 60

**if** remain\_sec **<=** 9:

            #*remain\_sec = "00" # dynamic type occuring "int" to "string" auto-conversion*

            #*remain\_sec = "0" + str(remain\_sec) # dynamic type occuring "int" to "string" auto-conversion*

            remain\_sec = f"0{remain\_sec}"   #*dynamic type occuring "int" to "string" auto-conversion*

        #*can\_vas.itemconfig() is like Label.coonfig()*

        can\_vas**.itemconfig**(timer\_text, text = f"{remain\_min}:{remain\_sec}")

        winDow**.after**(10, count\_down, count -1 )

**elif**  count **==** 0:

**start\_timer**()

**28.7 Adding Checkmarks and Resetting the Application**

To reset the timer we have to deactivate ***after()***, so we give it a name timer and when reset is clicked ***cancel\_after()*** will deactivate ***after()***. To do all these the timer itself need to be global so timer = None is been set.

Add the checkmark

**def** **count\_down**(count):

**global** tiMer

**if** count **>** 0:

        remain\_min = math**.floor**(count/60)

        remain\_sec = count % 60

**if** remain\_sec **<=** 9:

            #*remain\_sec = "00" # dynamic type occuring "int" to "string" auto-conversion*

            #*remain\_sec = "0" + str(remain\_sec) # dynamic type occuring "int" to "string" auto-conversion*

            remain\_sec = f"0{remain\_sec}"   #*dynamic type occuring "int" to "string" auto-conversion*

        #*can\_vas.itemconfig() is like Label.coonfig()*

        can\_vas**.itemconfig**(timer\_text, text = f"{remain\_min}:{remain\_sec}")

        tiMer = winDow**.after**(5, count\_down, count -1 )

**else**:

**global** check\_text

**if** reps % 2:

            check\_text += '✔'

            check**.config**(text = check\_text)

**start\_timer**()

TIMER RESET

#*---------------------------- TIMER RESET ------------------------------- #*

**def** **resEt**():

    winDow**.after\_cancel**(tiMer)

    can\_vas**.itemconfig**(timer\_text, text = "00:00")

    heading**.config**(text = "Timer")

    check**.config**(text = '')

Practiced version

**import**  tkinter

**import** math

#*---------------------------- CONSTANTS ------------------------------- #*

PINK = "#e2979c"

RED = "#e7305b"

GREEN = "#9bdeac"

YELLOW = "#f7f5dd"

FONT\_NAME = "Courier"

WORK\_MIN = 25

SHORT\_BREAK\_MIN = 5

LONG\_BREAK\_MIN = 20

tiMer = **None**

check\_text = ' '

#*---------------------------- TIMER RESET ------------------------------- #*

**def** **resEt**():

    winDow**.after\_cancel**(tiMer)

    can\_vas**.itemconfig**(timer\_text, text = "00:00")

    heading**.config**(text = "Timer")

    check**.config**(text = '')

#*---------------------------- TIMER MECHANISM ------------------------------- #*

reps = 0

    #*To invoke count\_down() when button is clicked*

**def** **start\_timer**():

**global** reps

    reps += 1

    work\_time = WORK\_MIN \* 60

    short\_break = SHORT\_BREAK\_MIN \* 60

    long\_break = LONG\_BREAK\_MIN \* 60

**if** (reps **>** 0) **and** (reps % 8 **==** 0):

**count\_down**(long\_break)

        heading**.config**(text = "LONG Break", fg =  RED)

        reps = 0

**elif** (reps **>** 0) **and** (reps % 2 **==** 0):

**count\_down**(short\_break)

        heading**.config**(text = "SHORT Break", fg =  PINK)

**else**:

**count\_down**(work\_time)

        heading**.config**(text = "WORK time", fg =  GREEN)

**print**(reps)

#*---------------------------- COUNTDOWN MECHANISM ------------------------------- #*

**def** **count\_down**(count):

**global** tiMer

**if** count **>** 0:

        remain\_min = math**.floor**(count/60)

        remain\_sec = count % 60

**if** remain\_sec **<=** 9:

            #*remain\_sec = "00" # dynamic type occuring "int" to "string" auto-conversion*

            #*remain\_sec = "0" + str(remain\_sec) # dynamic type occuring "int" to "string" auto-conversion*

            remain\_sec = f"0{remain\_sec}"   #*dynamic type occuring "int" to "string" auto-conversion*

        #*can\_vas.itemconfig() is like Label.coonfig()*

        can\_vas**.itemconfig**(timer\_text, text = f"{remain\_min}:{remain\_sec}")

        tiMer = winDow**.after**(5, count\_down, count -1 )

**else**:

**global** check\_text

**if** reps % 2:

            check\_text += '✔'

            check**.config**(text = check\_text)

**start\_timer**()

#*---------------------------- UI SETUP ------------------------------- #*

winDow = tkinter**.Tk**()

winDow**.minsize**(height = 300, width = 420)

winDow**.title**("Pomdoro GUI app")

#*Back ground color*

winDow**.config**(bg = YELLOW)

winDow**.config**(padx =100, pady = 50)

#*including image*

    #*PhotoImage is needed to attach image to tkinter*

pic\_tomato = tkinter**.PhotoImage**(file= "./tomato.png")

    #*crfreating Canvas for image: Its like a artist canvas, it allows layering*

can\_vas = tkinter**.Canvas**(width = 220, height = 250, bg = YELLOW, highlightthickness= 0)

            #*highlightthickness= 0 to disable canvas border*

    #*attaching image to canvas. "bg" for background color, "image" to insert image*

    #*must include x, y position*

can\_vas**.create\_image**(110, 125, image=pic\_tomato)

    #*wont show unless use a geometry manager: pack(), place() or grid()*

#*Grid is relative. To center the position  padding added to window*

can\_vas**.grid**(column = 1, row = 1)

#*Adding text top of image. Modify the text, use "fill" for color*

timer\_text = can\_vas**.create\_text**(110, 135, text = "00:00", fill = "white", font = (FONT\_NAME, 20, "bold"))

#*buttons and griding*

#*fg - foreground color/ text color , bg back-ground color*

heading = tkinter**.Label**(text = "Timer", bg = YELLOW, fg =  GREEN, font = (FONT\_NAME, 36, "bold") )

heading**.grid**(column = 1, row = 0)

#*# "bd" or "borderwidth" for border thickness*

button\_start = tkinter**.Button**(text = "Start", bg = RED, fg =  "white",  borderwidth = 0, font = (FONT\_NAME, 12, "bold") )

    #*start\_timer will invoke count\_down() when button is clicked*

button\_start**.config**(command = start\_timer)

button\_start**.grid**(column = 0, row = 2)

button\_reset = tkinter**.Button**(text = "Reset", bg = RED, fg =  "white", bd = 0, font = (FONT\_NAME, 12, "bold") )

    #*resEt() will invoke  when button is clicked*

button\_reset**.config**(command = resEt)

button\_reset**.grid**(column = 2, row = 2)

check = tkinter**.Label**(text = check\_text, bg = YELLOW, fg = GREEN ,font = (FONT\_NAME, 12))

check**.grid**(column = 1, row = 3)

winDow**.mainloop**()

#*python pomodoro\_main.py*

More Details

### Countdown Mechanism

### 1

* Event driven : GUI keeps checking on the events and when it detects the event(click etc), it's ready to react.'window.mainloop()' enables it. It loops through every millisecond to check a new event.  
  -if we use another 'while loop' inside 'window.mainloop', it won't be able to reach the mainloop and it won't even launch the app.

window.mainloop()

### 2

* .after()  
  Built-in method. It executes a command after a time delay; it makes the timer function in pomodoro app.  
  ms: in milleseconds. 1sec = 1000 milliseconds  
  window.after(1000, count\_down, count - 1)  
  -> wait for 1000 milliseconds (1sec), call function count\_down(function itself), passing in count(argument) -1

### 3

* Function recursion  
  to loop through a function without having to use while loop, we can use recursion of a function. Function calling itself inside the execution part. Basically looping through it. It has to have 'if statement' as a flag.  
  e.g. count\_down function inside .after( count\_down )

def count\_down(count):

if count > 0:

global timer

timer = window.after(1000, count\_down, count - 1)

timer is global variable holds None, to be used in another part.  
if count > 0:  
global timer  
timer = window.after(1000, count\_down, count - 1)

### 4

* Format the time  
  math module, floor returns largest whole number that is <= x  
  e.g. 4.8 -> 4  
  seconds: using modulo

count\_min = math.floor(count / 60)

count\_sec = count % 60

if count\_sec < 10:

count\_sec = f"0{count\_sec}"

canvas.itemconfig(timer\_text, text=f"{count\_min}:{count\_sec}")

### 5

* Count marks  
  Every two reps, tick marks every even number reps (Work-Break) 2nd, 4th, 6th ... math.floor(reps/2)

def count\_down(count):

else:

start\_timer()

mark = " "

work\_sessions = math.floor(reps/2)

for i in range(work\_sessions):

mark += "✓"

tick.config(text=mark)

### Start Timer

is responsible for calling countdown function. We have three timers; short break, long break and work

def start\_timer():

global reps

reps += 1

work\_sec = WORK\_MIN \* 60

short\_break\_sec = SHORT\_BREAK\_MIN \* 60

long\_break\_sec = LONG\_BREAK\_MIN \* 60

if reps % 8 == 0:

count\_down(long\_break\_sec)

title.config(text="Break", fg=RED)

elif reps % 2 == 0:

count\_down(short\_break\_sec)

title.config(text="Break", fg=PINK)

else:

count\_down(work\_sec)

title.config(text="Work")

* Reset Timer  
  -after\_cancel(): cancel waits, that we set previously.  
  -We made variable 'timer' as Global to be able to use it in both function 'reset\_timer()' and 'start\_timer()' function, but we assigned None to it.

timer = None

timer = window.after(1000, count\_down, count - 1)

def reset\_timer():

window.after\_cancel(timer)

Instructors solution

**from** tkinter **import** \*

**import** math

#*---------------------------- CONSTANTS ------------------------------- #*

PINK = "#e2979c"

RED = "#e7305b"

GREEN = "#9bdeac"

YELLOW = "#f7f5dd"

FONT\_NAME = "Courier"

WORK\_MIN = 25

SHORT\_BREAK\_MIN = 5

LONG\_BREAK\_MIN = 20

reps = 0

timer = **None**

#*---------------------------- TIMER RESET ------------------------------- #*

**def** **reset\_timer**():

    window**.after\_cancel**(timer)

    canvas**.itemconfig**(timer\_text, text="00:00")

    title\_label**.config**(text="Timer")

    check\_marks**.config**(text="")

#*---------------------------- TIMER MECHANISM ------------------------------- #*

**def** **start\_timer**():

**global** reps

    reps += 1

    work\_sec = WORK\_MIN \* 60

    short\_break\_sec = SHORT\_BREAK\_MIN \* 60

    long\_break\_sec = LONG\_BREAK\_MIN \* 60

**if** reps % 8 **==** 0:

**count\_down**(long\_break\_sec)

        title\_label**.config**(text="Break", fg=RED)

**elif** reps % 2 **==** 0:

**count\_down**(short\_break\_sec)

        title\_label**.config**(text="Break", fg=PINK)

**else**:

**count\_down**(work\_sec)

        title\_label**.config**(text="Work", fg=GREEN)

#*---------------------------- COUNTDOWN MECHANISM ------------------------------- #*

**def** **count\_down**(count):

    count\_min = math**.floor**(count / 60)

    count\_sec = count % 60

**if** count\_sec **<** 10:

        count\_sec = f"0{count\_sec}"

    canvas**.itemconfig**(timer\_text, text=f"{count\_min}:{count\_sec}")

**if** count **>** 0:

**global** timer

        timer = window**.after**(1000, count\_down, count - 1)

**else**:

**start\_timer**()

        marks = ""

        work\_sessions = math**.floor**(reps/2)

**for** \_ **in** **range**(work\_sessions):

            marks += "✔"

        check\_marks**.config**(text=marks)

#*---------------------------- UI SETUP ------------------------------- #*

window = **Tk**()

window**.title**("Pomodoro")

window**.config**(padx=100, pady=50, bg=YELLOW)

title\_label = **Label**(text="Timer", fg=GREEN, bg=YELLOW, font=(FONT\_NAME, 50))

title\_label**.grid**(column=1, row=0)

canvas = **Canvas**(width=200, height=224, bg=YELLOW, highlightthickness=0)

tomato\_img = **PhotoImage**(file="tomato.png")

canvas**.create\_image**(100, 112, image=tomato\_img)

timer\_text = canvas**.create\_text**(100, 130, text="00:00", fill="white", font=(FONT\_NAME, 35, "bold"))

canvas**.grid**(column=1, row=1)

start\_button = **Button**(text="Start", highlightthickness=0, command=start\_timer)

start\_button**.grid**(column=0, row=2)

reset\_button = **Button**(text="Reset", highlightthickness=0, command=reset\_timer)

reset\_button**.grid**(column=2, row=2)

check\_marks = **Label**(fg=GREEN, bg=YELLOW)

check\_marks**.grid**(column=1, row=3)

window**.mainloop**()

#*python instructor\_solution\_main.py*