Process
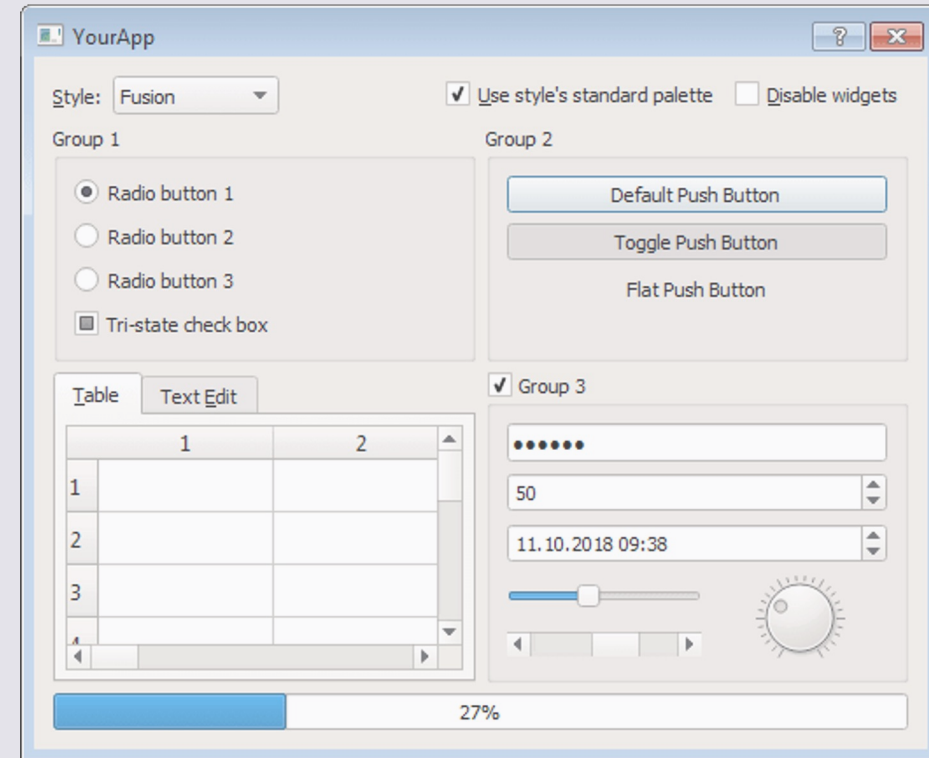
Your code should…

- Allow the user to enter important dates into a text file
- Output the number of days until each event by using python's Tkinter and datetime modules

Python's Tkinter module

The Tkinter module us a ser if tools that python programmers use for displaying graphics and getting input from the user. This is instead of running the program in a shell, and so the window can be designed and styled by the programmer

It allows the programmer to create a GUI (Graphical User Interface- a visible part of program that you can interact with). Examples of GUI's include icons and menus for applications on a smartphone.
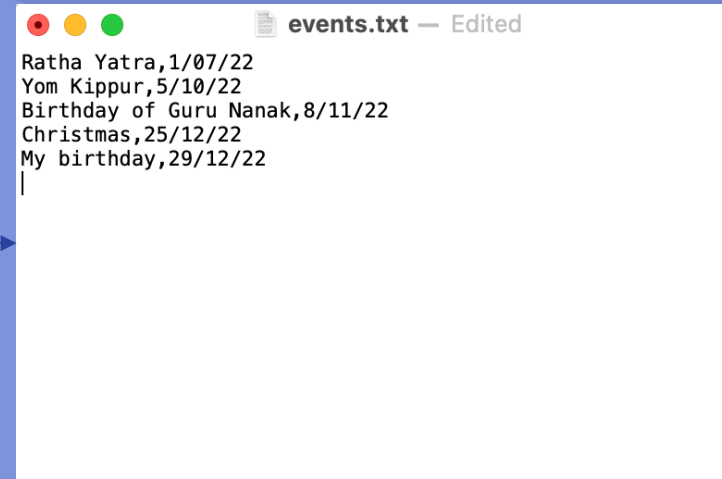
Tkinter allows you to build a GUI by using what we call widgets (packs of ready made code). These can create pop-up windows, buttons, menus, sliders and so on (above is an example of what Tkinter can allow you to create)

Step 1
Creating the program and the text file

Create the text file by opening a new IDLE file and typing in your big events, in the same format as pictured (with the name of the event first, separated by a comma, and the date written in the form: day/month/year). You should then save this file as events.txt

Create another new idle window, but this time save as a python file called familyCountdownCalendar.py

events.txt — Edited

```
Ratha Yatra,1/07/22
Yom Kippur,5/10/22
Birthday of Guru Nanak,8/11/22
Christmas,25/12/22
My birthday,29/12/22
```

*familyCountdownCalendar .py - /Users/   Desktop/Cair 4 Youth resources/familyCo...

Step 2

Importing the modules

Line 1- Tkinter will be used to build a very simple GUI

Line 2- datetime will make it easy to do calculations using the dates entered in the text file by the user

```
1 from tkinter import Tk, Canvas
2 from datetime import date, datetime
```

Subroutines

Subroutines are sets of instructions designed to perform a frequently used operation within a program.

```
1
2 def greeting():
3     print("Hello World!")
4     print("How are you today?")
5
6
7 greeting()
8
```

Hello World!
How are you?

Subroutines can store code and will only be run when 'called'

There are two main types of subroutine: procedures and **functions**.

Procedures are not required to return a value, whereas **functions must return a value.**

Subroutines are great ways of writing more maintainable code and leads to more structured, organized and understandable programs.

Lists

An list is a container which can hold a number of items of the same type.

Each item stored in an list is called an element, and its index is the position where it is within the array. Remember that python starts counting from 0, so the first element is at [0] not [1]

If you have a list of items (for example car names) storing the values individually means that it can be had to find a specific value (especially if you want to store 300 cars not 3)

We will use lists in our program to store information that we will read from the external file

```python
car1 = "Audi"
car2 = "BMW"
car3 = "Volvo"
```

```python
cars = ["Audi", "BMW", "Volvo"]
print(cars)
```

Step 3
Reading the text file

Line 4- a function called get_events() should be created, which will read and store all of the important events from the text file

Line 5- inside the function will be an empty list that will store the events when the file has been read. This list should be called list_events[ ]

```
4 def get_events():
5     list_events = []
```

Step 4
Opening the text file

Line 6- will open the file so that the program can read it

The open() function in python has to be written with the name of the text file in quotation marks every time the user wants to access information from the file.

```python
4 def get_events():
5     list_events = []
6     with open('events.txt') as file:
```

Loops

A loop is a sequence of instructions that is continually repeated until a certain condition is reached.

In Python there are two main loops: 'WHILE Loops' and 'FOR Loops'

While Loops are condition controlled and will repeat until their condition is false.

```python
condition = True
while condition:
    print("Repeating...")

    print("Finish loop?")
    finished = input()

    if finished == "Y":
        condition = False
```

```
Repeating...
Finish loop?
N
Repeating...
Finish loop?
N
Repeating...
Finish loop?
N
Repeating...
Finish loop?
Y
```

For loops are count controlled and will repeat a set number of times.

```python
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

Step 5

Adding a loop

Line 7- this for loop will contain the code for bringing in the information into your program. This loop will be run for every line in the events.txt file.
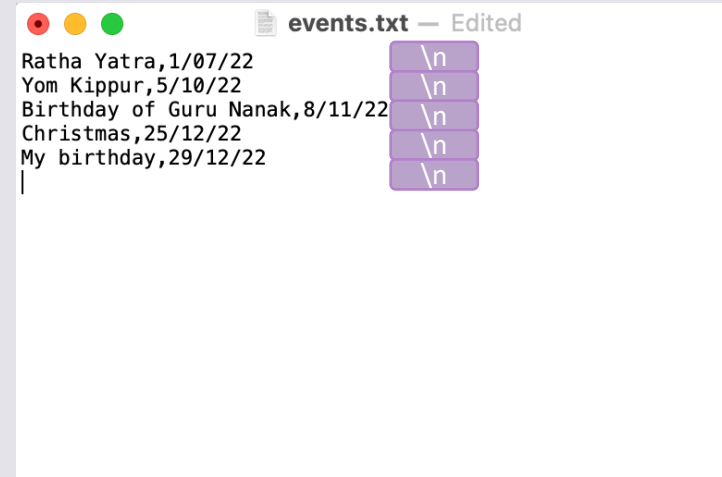
For example if you have 5 different events listed in your text file, then the loop will repeat 5 times

```
4 def get_events():
5     list_events = []
6     with open('events.txt') as file:
7         for line in file:
```

strip ( ) method

Python recognizes that you have used a new line between each element, and so sees this as a "newline" character, which must be gotten rid of, although not visible to the user. This is done using the line.rstrip('\n') method in python, which will essentially move this invisible character from the end of each of the lines in the text file

```
line = line.rstrip('\n')
```

events.txt — Edited

Ratha Yatra,1/07/22   \n
Yom Kippur,5/10/22   \n
Birthday of Guru Nanak,8/11/22   \n
Christmas,25/12/22   \n
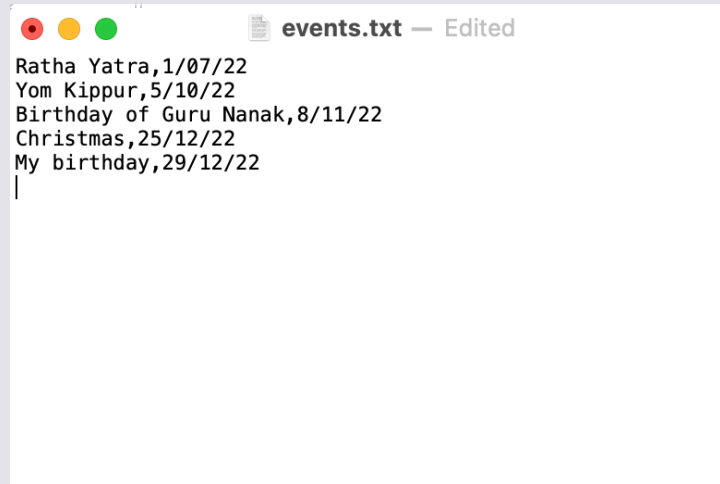My birthday,29/12/22   \n

Step 6

Removing the invisible character

Line 8- exactly as explained in the previous slide, the new line character will be removed from the end of each line in the text file

```
4 def get_events():
5     list_events = []
6     with open('events.txt') as file:
7         for line in file:
8             line = line.rstrip('\n')
```

cair
4 YOUTH

line.split ( ) method

The line.split() method will split each line of the text file at the comma. The parts before and after the comma will now become two separate items that can be stored


```
events.txt — Edited
Ratha Yatra,1/07/22
Yom Kippur,5/10/22
Birthday of Guru Nanak,8/11/22
Christmas,25/12/22
My birthday,29/12/22
```

Step 7

Store the event's details

At this point a variable called line holds the information about each event as one string, such as My birthday, 29/12/22

Line 9- the split() command will then be used as it was in the previous slide to break the string into two parts, and store in a new list called current_event

```python
4 def get_events():
5     list_events = []
6     with open('events.txt') as file:
7         for line in file:
8             line = line.rstrip('\n')
9             current_event = line.split(',')
```

Step 8
Using datetime

The current_event list stores both the name of the event as well as the date and time. We will use the code in lines 10 and 11 to convert the second item in the list from a string, to the date and time data type, so that we can use the datetime module to do calculations

Line 10- turns the second item in the list from a string to a date

Line 11- set the second item in the list to be the date of the event

```python
4  def get_events():
5      list_events = []
6      with open('events.txt') as file:
7          for line in file:
8              line = line.rstrip('\n')
9              current_event = line.split(',')
10             event_data = datetime.strptime(current_event[1], '%d/%m/%y').date()
11             current_event[1] = event_data
```

Step 9

Adding the event to the list

The current_event list holds two things: the name of the event (as a string) and the date of the event.

Line 12- adds the current_event to the list of events and, after this code is run, the program will loop back to line 8 and continue to look at the next line in the file

Line 13- after each line in the file has been looked at, the subroutine will return the list list_events and the for loop will not repeat again, as there will be no more lines to look at

```python
4  def get_events():
5      list_events = []
6      with open('events.txt') as file:
7          for line in file:
8              line = line.rstrip('\n')
9              current_event = line.split(',')
10             event_data = datetime.strptime(current_event[1], '%d/%m/%y').date()
11             current_event[1] = event_data
12             list_events.append(current_event)
13     return list_events
```

Step 10

Count the days

Line 15- a function is created that will count the number of days between the two dates. This is made easy by the datetime module as it can subtract the dates from one another.

Line 16- a variable called time_between holds the number of days between the two dates and this result is stored as a string

```
15  def days_between_dates(date1, date2):
16      time_between = str(date1 - date2)
```

Step 11

Returning the number of days

Line 17- only the number at the beginning of the time_between variable is important, as it will be formatted like 27 days 0:00:00. The split function can therefore be used to get just the start of the string as can be seen

Line 18- we only want to return the first part of that split line, therefore we have put [0] in the square brackets to access the first element (the number of days in the number_of_days variable)

```python
15  def days_between_dates(date1, date2):
16      time_between = str(date1 - date2)
17      number_of_days = time_between.split(' ')
18      return number_of_days[0]
```

Step 12

Creating the canvas

```
20 root = Tk()
```

Line 20- Uses Tkinter's root widget to create a window

```
21 c = Canvas(root, width=800, height=800, bg="black")
```

Line 21- the canvas widget is used to make a blank rectangle that text and graphics can be added to. This canvas will be 800 pixels wide and 800 pixels high in size, and black in colour

```
22 c.pack()
```

Line 22- the canvas is then "packed" into the Tkinter window

```
23 c.create_text(100, 50, anchor="w", fill='yellow', font="Arial 28 bold underline", text="My Countdown Calendar")
```

Line 23- the text for the title is then created- will be in the font Arial and underlined, and yellow in colour

What the canvas will look like…

The countdown to the important dates will be in orange below the title…

My Countdown Calendar

```
root = Tk()
c = Canvas(root, width=800, height=800, bg="black")
c.pack()
c.create_text(100, 50, anchor="w", fill='yellow', /
                font="Arial 28 bold underline", /
                text="My Countdown Calendar")
```

Step 13

Get the events

Now that all the functions have been written, we can call them in our main code

Line 25- stores the list of calander events in a variable called events after running the get_events() function

Line 26- uses the datetime module to get todays date and stores it in a variable called today

```
25 events = get_events()
26 today = date.today()
```

Step 14

Display the results

Line 29- We need to display the results for every event in the list, and so a for loop is used

Line 30– gets the name of the event

Line 32- creates a string to hold what we want to show on the screen

Line 31- uses the days_between_dates() function to calculate the number of days between the event and todays date

Line 33- displays the text on the screen at position vertical_space

```
29  for event in events:
30      event_name = event[0]
31      days_until = days_between_dates(event[1], today)
32      display = 'It is %s days until %s' % (days_until, event_name)
33      c.create_text(100,vertical_space,anchor='w', fill='orange',font='Arial 28 bold', text=display)
34
```

Vertical_space

This variable is created in order to avoid the text being stacked on top of each other. It increases its value every time the program goes through the for , loop which will mean that each time the code is run the text will be spaced out further along the screen

The code for the vertical_space variable should be inserted in the following places...

```
27 vertical_space = 100
28
29 for event in events:
30     event_name = event[0]
31     days_until = days_between_dates(event[1], today)
32     display = 'It is %s days until %s' % (days_until, e
33     c.create_text(100,vertical_space,anchor='w', fill='
34
35     vertical_space = vertical_space + 30
36
```

The final code and what it should look like when run...

cair 4 YOUTH

```python
1  from tkinter import Tk, Canvas
2  from datetime import date, datetime
3
4  def get_events():
5      list_events = []
6      with open('events.txt') as file:
7          for line in file:
8              line = line.rstrip('\n')
9              current_event = line.split(',')
10             event_data = datetime.strptime(current_event[1], '%d/%m/%y').date()
11             current_event[1] = event_data
12             list_events.append(current_event)
13     return list_events
14
15 def days_between_dates(date1, date2):
16     time_between = str(date1 - date2)
17     number_of_days = time_between.split(' ')
18     return number_of_days[0]
19
20 root = Tk()
21 c = Canvas(root, width=800, height=800, bg="black")
22 c.pack()
23 c.create_text(100, 50, anchor="w", fill='yellow', font="Arial 28 bold underline", text="My Countdown C
24
25 events = get_events()
26 today = date.today()
27 vertical_space = 100
28
29 for event in events:
30     event_name = event[0]
31     days_until = days_between_dates(event[1], today)
32     display = 'It is %s days until %s' % (days_until, event_name)
33     c.create_text(100,vertical_space,anchor='w', fill='orange',font='Arial 28 bold', text=display)
34
35     vertical_space = vertical_space + 30
```

```
My Countdown Calendar

It is 127 days until Ratha Yatra
It is 223 days until Yom Kippur
It is 257 days until Birthday of Guru Nanak
It is 304 days until Christmas
It is 308 days until My birthday
```