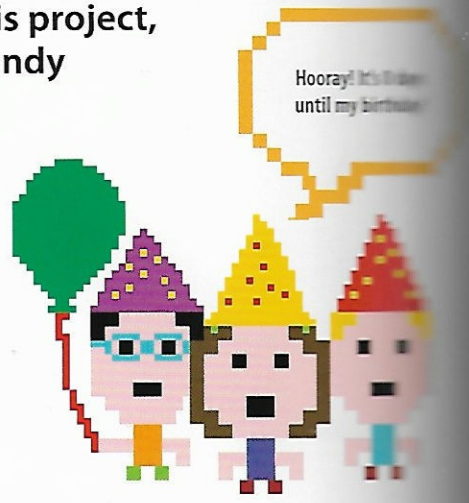
Countdown Calendar

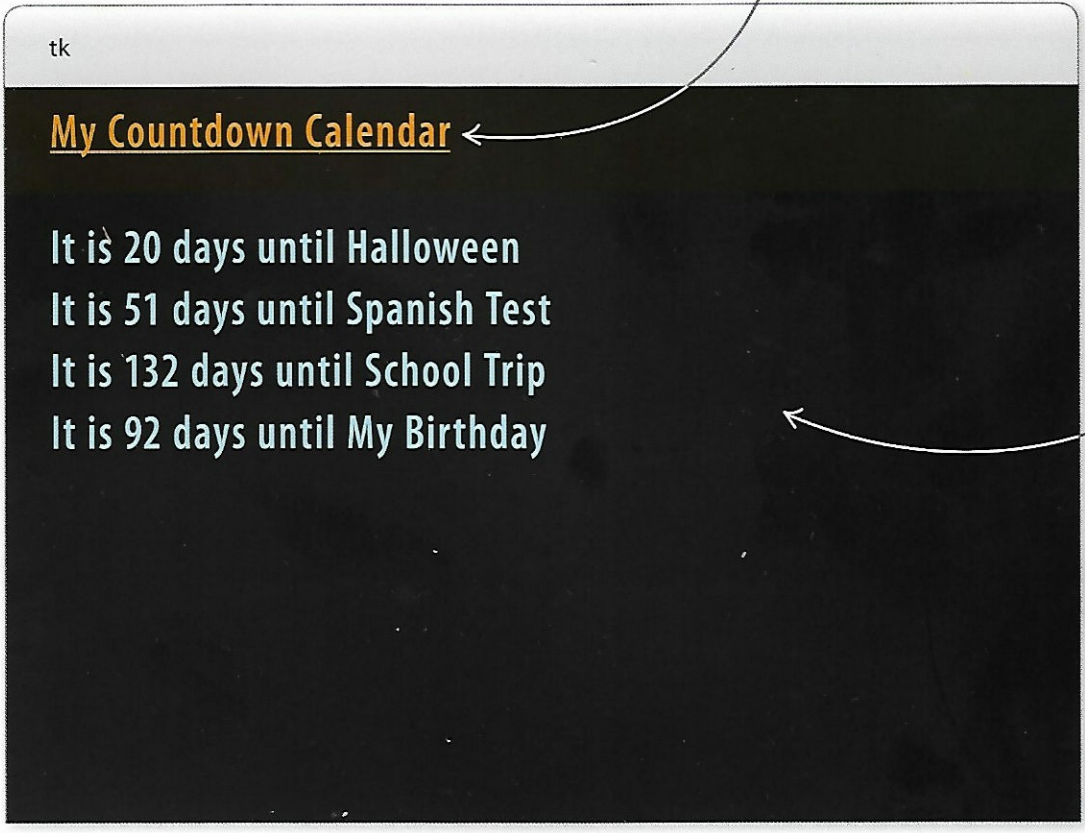
When you're looking forward to an exciting event, it helps to know how much longer you have to wait. In this project, you'll use Python's `Tkinter` module to build a handy program that counts down to the big day.



What happens

When you run the program it shows a list of future events and tells you how many days there are until each one. Run it again the next day and you'll see that it has subtracted one day from each of the "days until" figures. Fill it with the dates of your forthcoming adventures and you'll never miss an important day—or a homework deadline—again!

Give your calendar a personalized title.



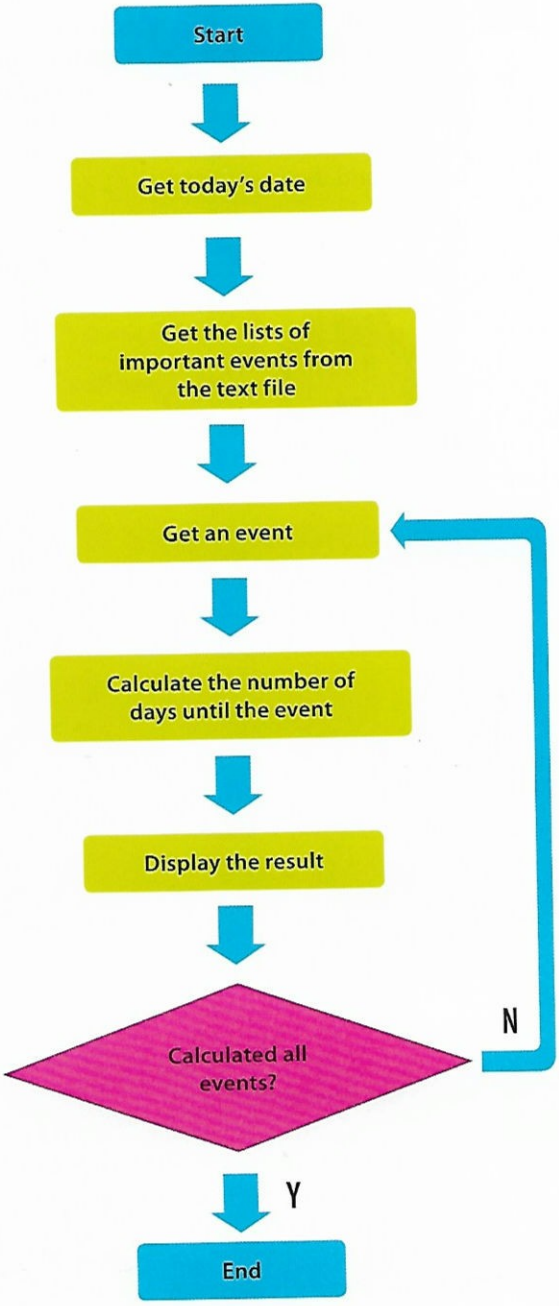
A small window pops up when you run the program with each event on a separate line.

How it works

The program learns about the important events by reading information from a text file—this is called "file input". The text file contains the name and date of each event. The code calculates the number of days from today until each event using Python's **`datetime`** module. It displays the results in a window created by Python's **`Tkinter`** module.

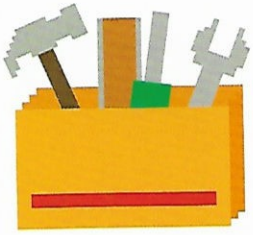
▽ **Countdown Calendar flowchart**

In this project, the list of important events is created separately from the code as a text file. The program begins by reading in all the events from this file. Once all the days have been calculated and displayed, the program ends.



▷ **Using Tkinter**

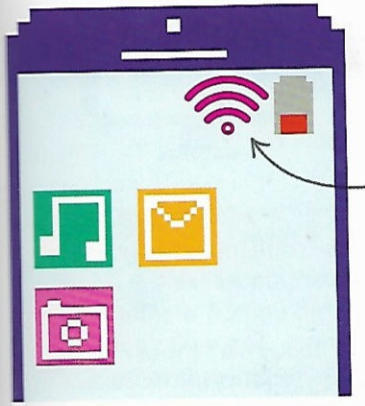
The `Tkinter` module is a set of tools that Python programmers use for displaying graphics and getting input from users. Instead of showing output in the shell, `Tkinter` can display results in a separate window that you're able to design and style yourself.



LINGO

Graphical user interface

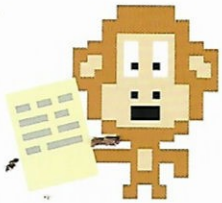
`Tkinter` is handy for creating what coders call a GUI (pronounced "gooey"). A GUI (graphical user interface) is the visible part of a program that a person interacts with, such as the system of icons and menus you use on a smartphone. `Tkinter` creates popup windows that you can add buttons, sliders, and menus to.



A smartphone GUI uses icons to show how strong the WiFi signal is and how much power the battery has.

Making and reading the text file

All the information for your Countdown Calendar must be stored in a text file. You can create it using IDLE.



1 Create a new file

Open a new IDLE file, then type in a few upcoming events that are important to you. Put each event on a separate line and type a comma between the event and its date. Make sure there is no space between the comma and the event date.

Type the date as day/month/year

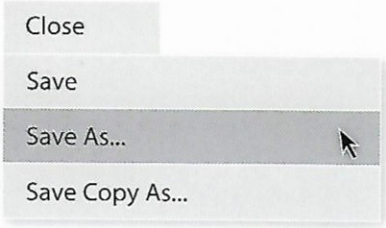
events.txt

```
Halloween,31/10/17
Spanish Test,01/12/17
School Trip,20/02/18
My Birthday,11/01/18
```

The name of the event comes first.

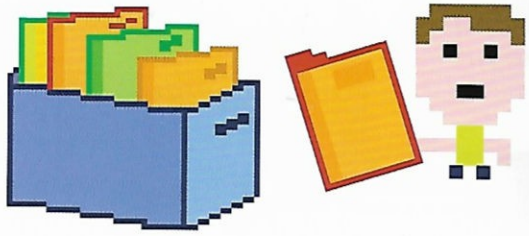
2 Save it as a text file

Next save the file as a text file. Click the File menu, choose Save As, and call the file "events.txt". Now you're ready to start coding the Python program.



3 Open a new Python file

You now need to create a new file for the code. Save it as "countdown_calendar.py" and make sure it's in the same folder as your "events.txt" file.



4 Set up the modules

This project needs two modules: **Tkinter** and **datetime**. **Tkinter** will be used to build a simple GUI, while **datetime** will make it easy to do calculations using dates. Import them by typing these two lines at the top of your new program.

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

Import the **Tkinter** and **datetime** modules.

5 Create the canvas

Now set up the window that will display your important events and the number of days until each one. Put this code beneath the lines you added in Step 4. It creates a window containing a "canvas"—a blank rectangle you can add text and graphics to.

This command packs the canvas into the `Tkinter` window.

Create a `Tkinter` window.

Create a canvas called `c` that is 800 pixels wide by 800 pixels high.

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

This line adds text onto the `c` canvas. The text starts at x = 100, y = 50. The starting coordinate is at the left (west) of the text.

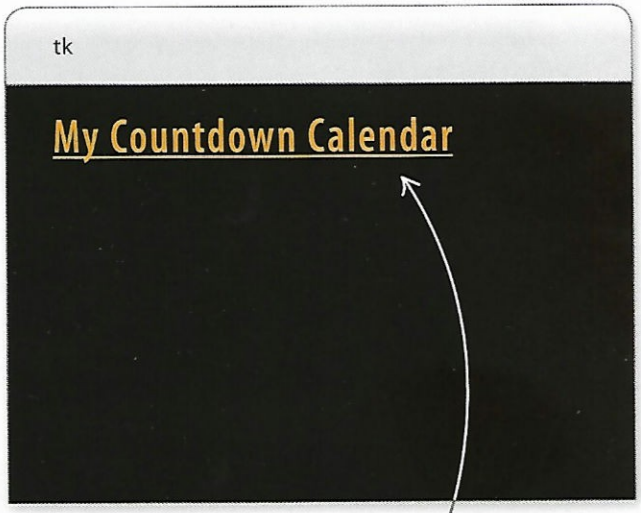
LINGO

Canvas

In `Tkinter`, the canvas is an area, usually a rectangle, where you can place different shapes, graphics, text, or images that the user can look at or interact with. Think of it like an artist's canvas—except you're using code to create things rather than a paintbrush!

6 Run the code

Now try running the code. You'll see a window appear with the title of the program. If it doesn't work, remember to read any error messages and go through your code carefully to spot possible mistakes.



You can change the colour by altering the `c.create_text()` line in the code.

7 Read the text file

Next create a function that will read and store all the events from the text file. At the top of your code, after importing the module, create a new function called `get_events`. Inside the function is an empty list that will store the events when the file has been read.

```
from datetime import date, datetime
def get_events():
    list_events = []
root = Tk()
```

Create an empty list called `list_events`.

8 Open the text file

This next bit of code will open the file called events.txt so the program can read it. Type in this line underneath your code from Step 7.

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

This line opens the text file.

9 Start a loop

Now add a `for` loop to bring information from the text file into your program. The loop will be run for every line in the events.txt file.

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

Run the loop for each line in the text file.

10 Remove the invisible character

When you typed information into the text file in Step 1, you pressed the enter/return key at the end of each line. This added an invisible "newline" character at the end of every line. Although you can't see this character, Python can. Add this line of code, which tells Python to ignore these invisible characters when it reads the text file.

```
    with open('events.txt') as file:
        for line in file:
            line = line.rstrip('\n')
```

Remove the newline character from each line.

The newline character is represented as `('\n')` in Python.

11 Store the event details

At this point, the variable called `line` holds the information about each event as a string, such as `Halloween,31/10/2017`. Use the `split()` command to chop this string into two parts. The parts before and after the comma will become separate items that you can store in a list called `current_event`. Add this line after your code in Step 10.

```
        for line in file:
            line = line.rstrip('\n')
            current_event = line.split(',')
```

Split each event into two parts at the comma.

EXPERT TIPS

Datetime module

Python's `datetime` module is very useful if you want to do calculations involving dates and time. For example, do you know what day of the week you were born on? Try typing this into the Python shell to find out.

Type your birthday in this format: year, month, day.

```
>>> from datetime import *
>>> print(date(2007, 12, 4).weekday())
1
```

This number represents the day of the week, where Monday is 0 and Sunday is 6. So December 4, 2007, was a Tuesday.

REMEMBER

List positions

When Python numbers the items in a list, it starts from 0. So the first item in your `current_event` list, "Halloween", is in position 0, while the second item, "31/10/2017", is in position 1. That's why the code turns `current_event[1]` into a date.

12 Using `datetime`

The event Halloween is stored in `current_event` as a list containing two items: "Halloween" and "31/10/2017". Use the `datetime` module to convert the second item in the list (in position 1) from a string into a form that Python can understand as a date. Add these lines of code at the bottom of the function.

Turns the second item in the list from a string into a date.

```
    current_event = line.split(',')
    event_date = datetime.strptime(current_event[1], '%d/%m/%y').date()
    current_event[1] = event_date
```

Set the second item in the list to be the date of the event.

13 Add the event to the list

Now the `current_event` list holds two things: the name of the event (as a string) and the date of the event. Add `current_event` to the list of events. Here's the whole code for the `get_events()` function.

```
def get_events():
    list_events = []
    with open('events.txt') as file:
        for line in file:
            line = line.rstrip('\n')
            current_event = line.split(',')
            event_date = datetime.strptime(current_event[1], '%d/%m/%y').date()
            current_event[1] = event_date
            list_events.append(current_event)
    return list_events
```

After this line is run, the program loops back to read the next line from the file.

After all the lines have been read, the function hands over the complete list of events to the program.

Setting the countdown

In the next stage of building Countdown Calendar you'll create a function to count the number of days between today and your important events. You'll also write the code to display the events on the `Tkinter` canvas.

14 Count the days

Create a function to count the number of days between two dates. The `datetime` module makes this easy, because it can add dates together or subtract one from another. Type the code shown here below your `get_events()` function. It will store the number of days as a string in the variable `time_between`.

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

The function is given two dates

This variable stores the result as a string.

The dates are subtracted to give the number of days between them.

15 Split the string

If Halloween is 27 days away, the string stored in `time_between` would look like this: `'27 days, 0:00:00'` (the zeros refer to hours, minutes, and seconds). Only the number at the beginning of the string is important, so you can use the `split()` command again to get to the part you need. Type the code highlighted below after the code in Step 14. It turns the string into a list of three items: `'27'`, `'days'`, `'0:00:00'`. The list is stored in `number_of_days`.


```
def days_between_dates(date1, date2):
    time_between = str(date1-date2)
    number_of_days = time_between.split(' ')
```

This time the string is split at each blank space.

16 Return the number of days

To finish off this function, you just need to return the value stored in position 0 of the list. In the case of Halloween, that's 27. Add this line of code to the end of the function.

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

The number of days between the dates is held at position 0 in the list.

17 Get the events

Now that you've written all the functions, you can use them to write the main part of the program. Put these two lines at the bottom of your file. The first line calls (runs) the `get_events()` function and stores the list of calendar events in a variable called `events`. The second line uses the `datetime` module to get today's date and stores it in a variable called `today`.

Use a backslash character if you need to split a long line of code over two lines.

Don't forget to save your work.

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

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

18 Display the results

Next calculate the number of days until each event and display the results on the screen. You need to do this for every event in the list, so use a `for` loop. For each event in the list, call the `days_between_dates()` function and store the result in a variable called `days_until`. Then use the `Tkinter create_text()` function to display the result on the screen. Add this code right after the code from Step 17.

The code runs for each event stored in the list of events.

Gets the name of the event.

Uses the `days_between_dates()` function to calculate the number of days between the event and today's date.

```
for event in events:
    event_name = event[0]
    days_until = days_between_dates(event[1], today)
    display = 'It is %s days until %s' % (days_until, event_name)
    c.create_text(100, 100, anchor='w', fill='lightblue', \
                  font='Arial 28 bold', text=display)
```

Creates a string to hold what we want to show on the screen.

This character makes the code go over two lines.

Displays the text on the screen at position (100, 100).

19 Test the program

Now try running the code. It looks like all the text lines are written on top of each other. Can you work out what's gone wrong? How could you solve it?

20 **Spread it out**

The problem is that all the text is displayed at the same location (100, 100). If we create a variable called **vertical_space** and increase its value every time the program goes through the **for** loop, it will increase the value of the y coordinate and space out the text further down the screen. That'll solve it!

My Countdown Calendar

It is 26 days until Halloween
It is 57 days until Spanish Test
It is 138 days until School Trip
It is 98 days until My Birthday

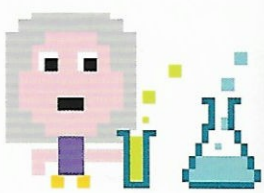
```
vertical_space = 100

for event in events:
    event_name = event[0]
    days_until = days_between_dates(event[1], today)
    display = 'It is %s days until %s' % (days_until, event_name)
    c.create_text(100, vertical_space, anchor='w', fill='lightblue', \
                  font='Arial 28 bold', text=display)

    vertical_space = vertical_space + 30
```

21 **Start the countdown!**

That's it—you've written all the code you need for Countdown Calendar. Now run your program and try it out.

Hacks and tweaks

Try these hacks and tweaks to make Countdown Calendar even more useful. Some of them are harder than others, so there are a few useful tips to help you out.



▷ **Repaint the canvas**

You can edit the background color of your canvas and really jazz up the look of the program's display. Change the **c = Canvas** line of the code.

```
c = Canvas(root, width=800, height=800, bg='green')
```

You can change the background color to any color of your choice.

Sort it!

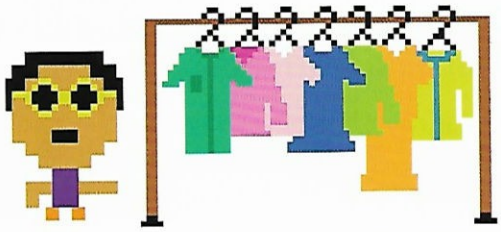
You can tweak your code so that the events get sorted into the order they'll be happening. Add this line of code before the `for` loop. It uses the `sort()` function to organize the events in ascending order, from the smallest number of days remaining to the largest.

```
vertical_space = 100
events.sort(key=lambda x: x[1])
for event in events:
```

Sort the list in order of days to go and not by the name of the events.

Restyle the text

Give your user interface a fresh new look by changing the size, color, and style of the title text.



Pick your favorite color.

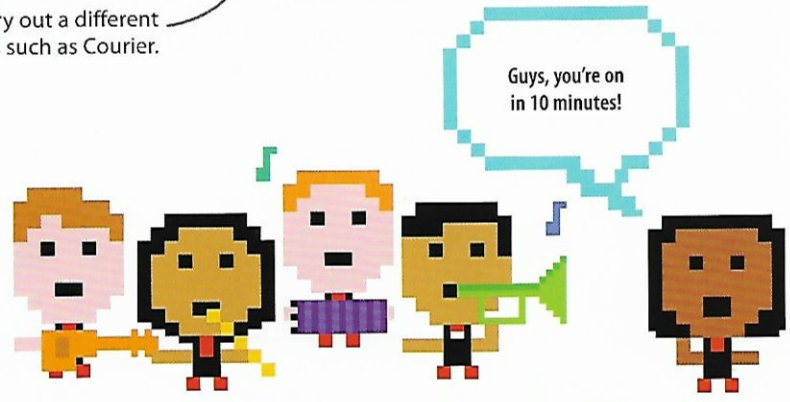
```
c.create_text(100, 50, anchor='w', fill='pink', font='Courier 36 bold underline', \
              text='Sanjay\'s Diary Dates')
```

Change the title too if you like.

Try out a different font, such as Courier.

Set reminders

It might be useful to highlight events that are happening really soon. Hack your code so that any events happening in the next week are shown in red.




```
for event in events:
    event_name = event[0]
    days_until = days_between_dates(event[1], today)
    display = 'It is %s days until %s' % (days_until, event_name)
    if (int(days_until) <= 7):
        text_col = 'red'
    else:
        text_col = 'lightblue'
    c.create_text(100, vertical_space, anchor='w', fill=text_col, \
                  font='Arial 28 bold', text=display)
```

The symbol <= means "is less than or equal to".

Display the text using the correct color.

The `int()` function changes a string into a number. For example, it turns the string '5' into the number 5.