# Problem 1 - Batch processing data files with a for loop

**Overview**

This problem is meant to simulate a common problem dealing with data files: batch processing. Batch processing involves using scripts to repeat a process with many data files, and one common task is generating a list of filenames that will be processed or saved to disk.

For this problem you will need to open a new Colab notebook.

## Part 1

* Create a new variable called basename that contains the text "Station".
* Create a new variable filenames that is an empty list.
* Use a print statement to check that they are set up correctly

## Part 2

* Using a for loop, iterate over the number range 0-20. Within the loop you should:
* Create a variable station that contains:
  + the text from the basename variable,
  + the number,
  + the file extension .txt
* Add the content of station to the filenames list, which should have following content in the end:

['Station\_0.txt', 'Station\_1.txt', 'Station\_2.txt', 'Station\_3.txt', 'Station\_4.txt', 'Station\_5.txt', 'Station\_6.txt', 'Station\_7.txt', 'Station\_8.txt', 'Station\_9.txt', 'Station\_10.txt', 'Station\_11.txt', 'Station\_12.txt', 'Station\_13.txt', 'Station\_14.txt', 'Station\_15.txt', 'Station\_16.txt', 'Station\_17.txt', 'Station\_18.txt', 'Station\_19.txt', 'Station\_20.txt']

* Check that the values of the last station is correct:

**assert** station**.**lower()**.**strip() **==** 'station\_20.txt', 'The value of the last station is not correct'

* Check that there are 21 values in the list:

**assert** len(filenames) **==** 21, 'The length of the list "filenames" should be 21'

## Part 3

Here, we ask you a few questions to make sure you have understood the concepts in this problem. Answer briefly in a few sentences using a Markdown cell .

You can also write any feedback or questions concerning this problem in this Markdown cell.

1. Is the concept of a loop clear to you? If not, what is difficult to understand?
2. Did you include comments in your code blocks? If not, please add them now

# Problem 2 - Classifying temperatures using conditional statements

**Overview**

This problem is designed to introduce you to the very common and useful concept of data classification. In this problem your aim is to classify daily temperatures (in degrees Celsius) stored in the temperatures list into four different classes:

* **Cold**: Temperatures of less than +4 degrees
* **Chilly**: Temperatures equal to or warmer than +4 degrees, but less than +8 degrees
* **Comfortable**: Temperatures equal to or warmer than +8 degrees, but less than +15 degrees
* **Warm**: Temperatures equal to or warmer than +15 degrees

To solve this problem, you should modify and fill in the missing parts in the following cells.

Data description

The data for this problem comprise a list of morning, midday and afternoon temperatures for April 2021 recorded between 9am and 9pm in Sheffield. The list contains 90 values since there are 3 values for each day (and April has 30 days). The first value of a given day represents morning, the second one is for the mid temperature, and the third one is for the maximum temperature. You can copy and paste this data into your Colab notebook.

temperatures **=** [0.5,3.075,7.4,-0.4,3.875,8.2,-0.9,5.1,10.7,6,10.25,12.9,9.3,10.425,13.1,8.6,9.85,12.8,4.5,6.6,9.3,2.3,5.55,10.2,2.5,5.775,9.7,0.8,7.55,12.5,5.8,10.15,15.2,6.6,9.325,12.7,10.2,12.8,15.7,5.9,11.95,17.6,7.7,14.75,20,8.2,14.5,20.7,8.6,14.175,19.4,8.4,12.075,15.3,4.2,9.7,14.2,5.8,10.475,16.7,4.4,10.1,16.6,6.6,10.25,16.3,7,9.6,14.5,5.9,9.975,14.4,4.1,8,12.1,5.7,8.575,12.8,4.4,7.15,9.4,5.9,7.25,8.8,5,8.9,13.6,3,11.2,17.4]

## Part 1

* Create four empty lists for the different temperature classes:
  + cold
  + chilly
  + comfortable
  + warm

Be sure to use these **exact** names for your lists and use print() to check they are empty.

# Test print for all lists (they should be empty at this point)

print(cold, chilly, comfortable, warm)

## Part 2

Iterate over the temperatures and add temperatures to the different temperature classes defined below:

* **Cold**: Temperatures of less than +4 degrees
* **Chilly**: Temperatures equal to or warmer than +4 degrees, but less than +8 degrees
* **Comfortable**: Temperatures equal to or warmer than +8 degrees, but less than +15 degrees
* **Warm**: Temperatures equal to or warmer than +15 degrees
* Check that your lists all have values.
* Please answer the following questions by using the following Python code format:

cold\_times **=** 'your-code-here'

print(f"In April 2021 it was cold {cold\_times} times.")

* How many times was it cold during the study period?
* How many times was it chilly during the study period?
* How many times was it comfortable during the study period?
* How many times was it warm during the study period?

## Part 3

Here, we ask a few questions to make sure you have understood the concepts in this problem. Answer shortly in a few sentences using a Markdown cell.

You can also write any feedback or questions concerning this problem in this Markdown cell.

1. Is the concept of conditional statements clear to you? If not, what is difficult to understand?
2. Did you include comments in your code blocks? If not, do it now :)

# Problem 3 - Allocating locations

**Overview**

The map below shows the locations of historic weather stations in the UK (blue points) that are more than 45 years old [1]. In this problem we are interested to find out whether the historical station network was equally distributed across the UK. We have the UK into four geographical zones (Northwest, Northeast, Southwest, Southeast) according the approximate centre point of the UK located at 54.093409,-2.89479 (latitude, longitude in decimal degrees).

A map of the world

Description automatically generated with medium confidence

[1]: The locations of weather stations were obtained from: [the UK Met Office](https://www.metoffice.gov.uk/research/climate/maps-and-data/uk-synoptic-and-climate-stations)

Below, we have given you the coordinates of the weather stations. The location of a single station is determined with a pair of latitude and longitude coordinates. The coordinates of all the stations are separated into two lists (lat and lon) and the names of the stations are in the stations list. From these lists, you would get, for example, the location of the first station by combining the latitude and longitude coordinates from the coordinate lists, and the name of that station from the stations list at index 0.

## Problem statement

In this problem your task is to print the names of weather stations located in different zones. You should also report the share of weather stations that allocated to each zone that could be used to evaluate if certain zone was over/under-represented 45 years ago.

To solve this problem, you should go through the following steps:

* Create four lists for geographical zones in the UK (i.e. north\_west, north\_east, south\_west, south\_east)
* Iterate over the values and determine to which geographical zone the station belongs
  + **Hint:** You should create a loop that iterates n times. The variable n should contain the number of stations we have.
  + You should use a conditional statement to find out if the latitude coordinate of a station is either North or South of the centre point of the UK (54.093409,-2.89479) **AND** if the longitude location is West or East of that centre point.
  + You should insert the name of the station into the correct geographical zone list
* Print out the names of stations in each geographical zone
* Calculate the percentage of all weather stations that fall in each zone.

## The data

Here, we provide the data you should use to solve Problem 3 (you can copy and paste these into your Colab notebook).

# historic weather station names

stations **=** ['Aberporth', 'Armagh', 'Ballypatrick Forest', 'Bradford', 'Braemar', 'Camborne', 'Cambridge NIAB', 'Cardiff Bute Park', 'Chivenor', 'Cwmystwyth', 'Dunstaffnage', 'Durham', 'Eastbourne', 'Eskdalemuir', 'Heathrow', 'Hurn', 'Lerwick', 'Leuchars', 'Lowestoft', 'Manston', 'Nairn', 'Newton Rigg', 'Oxford', 'Paisley', 'Ringway', 'Ross-on-Wye', 'Shawbury', 'Sheffield', 'Southampton', 'Stornoway Airport', 'Sutton Bonington', 'Tiree', 'Valley', 'Waddington', 'Whitby', 'Wick Airport', 'Yeovilton']

#Latitude coordinates of Weather stations

lat **=** [52.13914, 54.35234, 55.18062, 53.81341, 57.00612, 50.21782, 52.24501, 51.48783, 51.08865, 52.35817, 56.45054, 54.76786, 50.76167, 55.311, 51.47872, 50.7789, 60.13946, 56.37745, 52.48323, 51.34597, 57.593, 54.6699, 51.76073, 55.8455, 53.356, 51.91075, 52.79433, 53.38101, 50.89822, 58.21382, 52.8331, 56.49999, 53.25238, 53.17509, 54.48073, 58.45406, 51.00586]

# Longitude coordinates of Weather stations

lon **=** [-4.56999, -6.64866, -6.15336, -1.77234, -3.39635, -5.32656, 0.10196, -3.18728, -4.14743, -3.80198, -5.43859, -1.58455, 0.28543, -3.206, -0.44904, -1.83483, -1.18299, -2.86051, 1.72679, 1.33716, -3.82097, -2.78644, -1.2625, -4.42966, -2.279, -2.58441, -2.66329, -1.48986, -1.40839, -6.31772, -1.25, -6.8796, -4.53524, -0.52173, -0.62411, -3.0884, -2.64148]

# Cutoff values that correspond to the centroid of the UK

# North - South

north\_south\_cutoff **=** 54.09

# East-West

east\_west\_cutoff **=** -2.89

## Step 1

* Create four empty lists for the geographical zones in the UK. Use **exactly** these variable names:

north\_west

north\_east

south\_west

south\_east

## Step 2

* Count the total number of stations and store that value in the variable n.
* Create a print statement that prints out a statement stating the number of weather stations in 1978.

## Step 3

* Make a loop that iterates n times, and allocates stations to different geographical zones based on their coordinates.
* In the following cells, you should print out the correct station names for each geographical zone.
  + We have given you the correct number of stations as a hint for each zone, so it is easier for you to know whether you have correct answer.

## Step 6

* Print the percentage of stations in each geographical zone
  + Store the answers into variables:

north\_west\_share

north\_east\_share

south\_west\_share

south\_east\_share

* Print the results following format: "Northwest contains 99 % of all stations." (this is an example not a correct answer) using the code below.

# Print the information (you don't need to modify this)

# Note we are using f-strings here

# .0f rounds the decimal values to whole numbers

print(f"Northwest contains {north\_west\_share:.0f}% of all stations.")

print(f"Northeast contains {north\_east\_share:.0f}% of all stations.")

print(f"Southwest contains {south\_west\_share:.0f}% of all stations.")

print(f"Southeast contains {south\_east\_share:.0f}% of all stations.")

## Step 5

Here, we ask a few questions to make sure you have understood the concepts in this problem. Answer briefly with in few sentences using the Markdown cell below.

You can also write any feedback or questions concerning this problem in this Markdown cell.

1. How relevant do you think it is to calculate the proportion of stations in each zone? DO you have any alternative suggestions?
2. Is the concept of using the and operator and the difference between the if, elif, and else conditional statements clear to you? If not, what is difficult to understand?
3. Did you include comments in your code blocks? If not, please add them now :)

# Problem 4 - Nested for loops (optional)

**Warning:** This is a more difficult problem. You are welcome to try to do this task if you are up for a little challenge and have confidence in your programming skills. :)

In addition to having single for loops that iterate across some variable range, it is possible to *nest* for loops within one another.

Consider the example below:

**>>>** **for** char **in** 'dog':

**...** **for** char2 **in** 'cat':

**...** print (char, char2)

d c

d a

d t

o c

o a

o t

g c

g a

g t

Here, you can see that in the first pass through the first for loop, the value of char is d. Entering the inner (or nested) loop, char2 is set to c. After this, the output is written to the screen and since there are more letters to loop over in the inner for loop, the value of char2 will be updated upon the next pass. The second time through the inner loop, char2 is set to a while char remains d. Like this, the inner loop will run through all of the letters in cat for each time that char is updated in the outer loop. It doesn't take too much imagination to figure out this is a very useful concept.

For this problem you should do following:

1. Create a variable star with text "\*" and an empty string variable text. (Recall, you can created empty string variables by assigning "" as their value.
2. Use nested for loops and the variables above to produce the text formation below when print(text) is run at the end of your script.

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

1. Create a variable line with text "-" and an empty string variable flag.
2. Use nested for loops and the variables above to produce the text formation below when print(flag) is run at the end of your script. **Note**: You will need to use conditional statements to produce the desired output.

\*\*\*\*\*\*\*------------

\*\*\*\*\*\*\*------------

\*\*\*\*\*\*\*------------

-------------------

-------------------

All done!

Congrats! If you are here, you have finished all problems for week 3, even the extra one! Bad-ass.

Now just remember to upload your notebook to the Exercise 3 discussion thread.