Intro_Python_2022_Solutions

July 19, 2022

1 Exercises

You get a Series of Temperature Measurements from the MeteoSwiss Weather Station in Zollikofen. The data is for one day from 00:00 to 23:50 with a measurement every 10 minutes.

[ ]:
```
# This code snipped will read the data from the csv file and give you a list.␣
 ↪You don't need to understand everything
# that is happening yet.

import csv #we can use the csv module to read a csv file
# first we have to open the file
with open("/lhome/cra2022/cra2022_shared/temperatures_zollikofen.csv",␣
 ↪newline='') as file:
    reader = csv.reader(file) # then we use a reader object to read the content␣
 ↪of the file
    temperatures = next(reader) # Now we put the content of the reader into a␣
 ↪list
temperatures = [float(x) for x in temperatures] # The values are read as␣
 ↪strings. We have to convert them to float first
# Here we used "list comprehension" this is a handy, python specific way to␣
 ↪apply functions to a list
print(temperatures)
```

So first let's check if we have all the measurements we need to investigate the whole day. With 24h and 6 measurements per hour we would expect 144 values. How many are there in the list we've read from?

[ ]:
```
len(temperatures)
```

There is a value missing! It's infact the first value for 00:00 and it would be 2.949576240750525°C. Can you put it back into the list at the correct location?

[ ]:
```
temperatures.insert(0,2.949576240750525)
len(temperatures)
```

Now let's get a first impression of the values. Can you figure out what the minimum and maximum temperatures of that day were?

[ ]:
```python
max(temperatures)
min(temperatures)
```

Hmm.. our values look a bit strange. The minimum and maximum temperatures are quite close to each other and the numbers themselves look odd. For some reason someone stored the square roots of all the numbers! Before we can continue we have to fix that. Can you convert the numbers back to their original values?

[ ]:
```python
#Solution using a for loop
original_temperatures = []
for value in temperatures:
    original_temperatures.append(value**2)

#alternative solution using list comprehension:
original_temperatures = [value**2 for value in temperatures]

original_temperatures
```

The values still look a bit weird. We do not need such high precision. Can you round them to only 1 value after the decimal point?

[ ]:
```python
#Solution using a for loop
rounded_temperatures = []
for value in original_temperatures:
    rounded_temperatures.append(round(value,1))

#alternative solution using list comprehension:
rounded_temperatures = [round(value,1) for value in original_temperatures]

rounded_temperatures
```

Now that we have the original measurements we can look at the data again. We are actually interested in the fourth highest temperature of the day. What is it?

[ ]:
```python
sorted_temperature = sorted(rounded_temperatures, reverse = True)
sorted_temperature[3]

# You could also use the following line. But you have to be careful. This␣
 ↪function works "in-place"
# which means that it changes rounded_temperatures itself. This can be a␣
 ↪problem, for example if you think about the
# next exercise.

# rounded_temperatures.sort(reverse = True)[3]
```

10-minute temporal resolution is quite high. Hourly averages would be enough to continue working. Can you calculate a list containing these 24 hourly averages?

[ ]:

```python
# pure python implementation with all the things you have seen today
averages = []
window_size = 6
i = 0
while i < len(rounded_temperatures) - window_size + 1:
    window = rounded_temperatures[i : i + window_size]
    window_average = sum(window) / window_size
    averages.append(window_average)
    i += window_size

print(moving_averages)
```