Processor temperature

We have a temperature sensor in the processor of our company's server. We want to analyze the data provided to determinate whether we should change the cooling system for a better one. It is expensive and as a data analyst we cannot make decisions without a basis.

We provide the temperatures measured throughout the 24 hours of a day in a list-type data structure composed of 24 integers:

```
temperatures_C = [33,66,65,0,59,60,62,64,70,76,80,69,80,83,68,79,61,53,
50,49,53,48,45,39]
```

Goals

1. Treatment of lists
2. Use of loop or list comprenhention
3. Calculation of the mean, minimum and maximum.
4. Filtering of lists.
5. Interpolate an outlier.
6. Logical operators.
7. Print

Temperature graph

To facilitate understanding, the temperature graph is shown below. You do not have to do anything in this section. The test starts in **Problem**.

In [133]:

```
# import
import matplotlib.pyplot as plt
%matplotlib inline

# axis x, axis y
y = [33,66,65,0,59,60,62,64,70,76,80,81,80,83,90,79,61,53,50,49,53,48,45,39]
x = list(range(len(y)))

# plot
plt.plot(x, y)
plt.axhline(y=70, linewidth=1, color='r')
plt.xlabel('hours')
plt.ylabel('Temperature ºC')
plt.title('Temperatures of our server throughout the day')
```

```
---------------------------------------------------------------------
---------
ModuleNotFoundError                       Traceback (most recent c
all last)
<ipython-input-133-5d623b03114c> in <module>
      1 # import
----> 2 import matplotlib.pyplot as plt
      3 get_ipython().run_line_magic('matplotlib', 'inline')
      4
      5 # axis x, axis y

ModuleNotFoundError: No module named 'matplotlib'
```

Problem

If the sensor detects more than 4 hours with temperatures greater than or equal to 70ºC or any temperature above 80ºC or the average exceeds 65ºC throughout the day, we must give the order to change the cooling system to avoid damaging the processor.

We will guide you step by step so you can make the decision by calculating some intermediate steps:

1. Minimum temperature
2. Maximum temperature
3. Temperatures equal to or greater than 70ºC
4. Average temperatures throughout the day.
5. If there was a sensor failure at 03:00 and we did not capture the data, how would you estimate the value that we lack? Correct that value in the list of temperatures.
6. Bonus: Our maintenance staff is from the United States and does not understand the international metric system. Pass temperatures to Degrees Fahrenheit.

Formula: F = 1.8 * C + 32

web: https://en.wikipedia.org/wiki/Conversion_of_units_of_temperature (https://en.wikipedia.org/wiki/Conversion_of_units_of_temperature)

In [134]:

```
# assign a variable to the list of temperatures
temperatures_C = [33,66,65,0,59,60,62,64,70,76,80,81,80,83,90,79,61,53,50,49,5
3,48,45,39]

# 1. Calculate the minimum of the list and print the value using print()

minimum_temperature = min(temperatures_C)

print(minimum_temperature)
```

```
0
```

Expected output:

```
minimum = 0
```

In [135]:

```
# 2. Calculate the maximum of the list and print the value using print()
maximum_temperature = max(temperatures_C)

print(maximum_temperature)
```

```
90
```

Expected output:

```
maximum = 90
```

In [136]:

```
# 3. Items in the list that are greater than 70ºC and print the result
temperature_greaterthan_70 = []

for temperature in temperatures_C:
    if temperature > 70:
        temperature_greaterthan_70.append(temperature)

print("temperatures higher or equal than 70ºC", temperature_greaterthan_70)
```

```
temperatures higher or equal than 70ºC [76, 80, 81, 80, 83, 90, 79
]
```

Expected output:

```
temperatures higher or equal than 70ºC [70, 76, 80, 81, 80, 83, 90, 79]
```

In [137]:

```
# 4. Calculate the mean temperature throughout the day and print the result
mean_temperature = sum(temperatures_C) / len(temperatures_C)

print(mean_temperature)
```

```
60.25
```

Expected output:

```
mean = 60.25
```

In [138]:

```
# 5.1 Solve the fault in the sensor by estimating a value

temperature_at_300 = (temperatures_C[2]+temperatures_C[4])/2

print("Estimation of the temperature at 3:00 =", temperature_at_300)
```

```
Estimation of the temperature at 3:00 = 62.0
```

Expected output:

```
Estimation of the temperature at 3:00 = 62.0
```

In [139]:

```
# 5.2 Update of the estimated value at 03:00 on the list

temperatures_C[3] = temperature_at_300

print("Corrected temperatures after estimation:", temperatures_C)
```

```
Corrected temperatures after estimation: [33, 66, 65, 62.0, 59, 60
, 62, 64, 70, 76, 80, 81, 80, 83, 90, 79, 61, 53, 50, 49, 53, 48,
45, 39]
```

Expected output:

```
Corrected temperatures after estimation:  [33, 66, 65, 62.0, 59, 60, 62
, 64, 70, 76, 80, 81, 80, 83, 90, 79, 61, 53, 50, 49, 53, 48, 45, 39]
```

In [171]:

```
# Bonus: convert the list of ºC to ºFarenheit

def convert_to_farenheit(list):
    farenheit_list = []
    for x in list:
        new_farenheit = 1.8*x +32
        farenheit_list.append(new_farenheit)
    return farenheit_list

print("Temperatures in Fahrenheit Grades =", convert_to_farenheit(temperatures
_C))

temperatures_F = convert_to_farenheit(temperatures_C)
```

```
Temperatures in Fahrenheit Grades = [91.4, 150.8, 149.0, 143.60000
000000002, 138.2, 140.0, 143.60000000000002, 147.2, 158.0, 168.8,
176.0, 177.8, 176.0, 181.4, 194.0, 174.20000000000002, 141.8, 127.
4, 122.0, 120.2, 127.4, 118.4, 113.0, 102.2]
```

Expected output:

```
Temperatures in Fahrenheit Grades = [91.4, 150.8, 149.0, 143.6000000000
0002, 138.2, 140.0, 143.60000000000002, 147.2, 158.0, 168.8, 176.0, 177
.8, 176.0, 181.4, 194.0, 174.20000000000002, 141.8, 127.4, 122.0, 120.2
, 127.4, 118.4, 113.0, 102.2]
```

Take the decision

Remember that if the sensor detects more than 4 hours with temperatures greater than or equal to 70ºC or any temperature higher than 80ºC or the average was higher than 65ºC throughout the day, we must give the order to change the cooling system to avoid the danger of damaging the equipment:

- more than 4 hours with temperatures greater than or equal to 70ºC
- some temperature higher than 80ºC
- average was higher than 65ºC throughout the day If any of these three is met, the cooling system must be changed.

In [158]:

```
# Print True or False depending on whether you would change the cooling system
or not


def change_cooling_system(lst):
    for x in range(len(lst)-4):
        if lst[x] >= 70 and lst[x+1] >= 70 and lst[x+2] >= 70 and lst[x+3] >=
70 and lst[x+4] >= 70:
            return True
        if lst[x] > 80:
            return True
    if (sum(lst) / len(lst)) > 65:
        return True
    else:
        return False

print(change_cooling_system(temperatures_C))
```

```
True
```

Expected output:

```
True
```

Future improvements

1. We want the hours (not the temperatures) whose temperature exceeds 70ºC
2. Condition that those hours are more than 4 consecutive and consecutive, not simply the sum of the whole set. Is this condition met?
3. Average of each of the lists (ºC and ºF). How they relate?
4. Standard deviation of each of the lists. How they relate?

In [157]:

```
# 1. We want the hours (not the temperatures) whose temperature exceeds 70ºC
hours = []
for x in range(len(temperatures_C)):
    if temperatures_C[x] >= 70:
        hours.append(x)
print(hours)
```

```
[8, 9, 10, 11, 12, 13, 14, 15]
```

Expected output:

```
[8, 9, 10, 11, 12, 13, 14, 15]
```

In [168]:

```
# 2. Condition that those hours are more than 4 consecutive and consecutive, n
ot simply the sum of the whole set. Is this condition met?

example_list = [1,5,6,3,4,6,10,1,2,3,4,5]

def consecutive_hours(lst):
    is_it_consecutive = False
    for x in range(len(lst)-4):
        if (lst[x+4] == lst[x] + 4) and (lst[x+3] == lst[x] + 3) and (lst[x+2]
== lst[x] + 2) and (lst[x+1] == lst[x] +1):
            is_it_consecutive = True
    return is_it_consecutive

print(consecutive_hours(example_list))
```

```
True
```

Expected output:

```
True
```

In [176]:

```
# 3. Average of each of the lists (ºC and ºF). How they relate?

mean_C = sum(temperatures_C) / len(temperatures_C)
mean_F = sum(temperatures_F) / len(temperatures_F)

mean_C_to_F = 1.8*mean_C +32

print(mean_C, mean_F, mean_C_to_F)
```

```
62.833333333333336 145.1 145.10000000000002
```

Expected output:

```
62.833333333333336
145.1
145.1 145.10000000000002
```

In [13]:

```
# 4. Standard deviation of each of the lists. How they relate?
```

Expected output:

```
14.633485192833897
26.34027334710101
26.34027334710101 26.340273347101014
```