Hausarbeit

Semester: WS 16/17

Abgabetermin: 21 April

Nummer und Titel der Veranstaltung: 41647

Seminarleitung: Sebastian Gassner

Titel der Hausarbeit: Programm in Python

Mat.-Nr.: 79519

O Prüfungs-Nr.: Bitte Schein ausstellen für AAA

O Zertifikat Digital Humanities: -

O Studiengang: B.Sc Informatik

O Fachsemester: 1.

O Name/Vorname:Pioli Junior, Laercio

O Adresse: Neuburguerstrasse 124 OG.2 -Passau-Deustschland oder Avenida Daminhão Junqueira de souza, Stadt-São Lourenço, CEP 37470-000, Minas Gerais, Brasil

O E-Mail: laerciopiolijr@gmail.com

Wird vom Korrektor ausgefüllt:

Note:	**Punkte:**
Ort, Datum:	**Unterschrift:**

A short task analysis

The following Software is responsible for providing information of the hydrological water levels of some stations in Germany.

The software was developed in the Python3 Programming Language, and the data is indescribable through HTTP protocols where some URLs present their content.

It was developed in this software, as requested some functionalities as:

Check the list of the existing water stations.

What is the Minimum, Maximum, and Arithmetic Mean of water present in each station over a period of 15-Days past.

And it is also possible to check the results through graphics.

Functions

1- def load_stations(url_param,endpoint_param):

```
    This function receives a URL and an Endpoint and loads the data (stations) into
a memory variable in Python format for later use in the program.


  :param url_param {str}: Base URL address for downloading all data
  :param endpoint_param {str}:Complementary address of the URL that characterizes
the water stations.
      :return:data: Returns a List with all the stations present on the Web. The
entire program will use the saved data present in this file.
       :error: Offline page errors were handled
    try:
        url = url_param
        endpoint = endpoint_param
        response = urllib.request.urlopen(url + endpoint)
        body = response.read().decode("utf-8")
        data = json.loads(body)
    except:
        print("Website is offline, please try again later")
    return data
```

2- def stationen_auflisten(dados):

```
     This function returns a dictionary with all water stations, where the key is
its alphabetical position and value is the name of the station.

    :param dados {list}: Receives a list of dictionaries, where each dictionary is
a water station with several attributes (uuid, number, shortname, longname, km,
agency, longitude, latitude, water

    :return:Returns a dictionary in alphabetical order, which will aid the
presentation of item 2 [L] iste

    :error: No errors/difficult found

    data = dados
    new_dict = {item['longname']: item for item in data}
    allstations = {}
    count = 0
    for i in sorted(new_dict):
        allstations[count] = i
        count += 1
    return allstations
```

3- def searchinitialletter (initialletter, list_cidades):

```
    This function searches for the letters entered by the user in the list
of stations passed by parameter to see if that station exists.

    :param initialletter {str}: Letter entered by the user
    :param list_cidades {list}: List with all the stations that are in the data
document

    :return: Returns a list with all the stations with the name that the user
entered if it exists.

    :error: No errors/difficult found

    i = 0
    aux_stationfound = []
    while i < len(list_cidades):
        if initialletter in list_cidades[i]:
            aux_stationfound.append(list_cidades[i])
        i += 1
    if len(aux_stationfound) == 0:
        print("This station could not be identified, please try again")
    return aux_stationfound


4- def returnnumberid (namecidade,dados):

     This function receives as a parameter a name of a station and the total data
set and returns the identified station.

    :param namecidade {str}: The name of a particular station
    :param dados {list}: Total set of data read with the load_stations function

    :return:Returns a String with the corresponding station identifier.

    :error: No errors/difficult found

    number = None
    for i in dados:
        cidade_dados = i.get('longname')
        if namecidade == cidade_dados:
            number = i.get('number')
        else:
            False
    return number

5- def returnlistid(list_stationfound):

     This function receives as a parameter a list with the stations that were found
when the user searched the [E] tatistic (statistic).

    :param list_stationfound {list}:Receives the name of the stations found in the
data.

    :return: Returns a list with all the identifiers of the stations found

    :error: No errors/difficult found

    j = 0
    list_id = []
    while j < len(list_stationfound):
        id = returnnumberid(list_stationfound[j], dados)
        list_id.append(id)
        j += 1
    return list_id


6- def printdata(list):

     This function receives a list like this ['PASSAU ILZSTADT', 456.0, 608.0,
501.8209169054441] and applies the formatting required to be able to print the
data.

    :param list {list}:This list contains the values (station name, Minimo, Maximo
and AVG) that have already been calculated

    :return: No return

    :error: No errors/difficult found

    name = list[0]
    name15 = []
    minimum = float(list[1])
    maximum = float(list[2])
    avg = list[3]
    avg = ("%10.2f" % avg)
    minimum = ("%10.2f" % minimum)
    maximum = ("%10.2f" % maximum)
    checker = True
    while checker:
        if len(name) > 15:
            for i in name:
                letra = i
                if len(name15) < 15:
                    name15.append(letra)
            print(''.join(name15), "    ", minimum, "   ", maximum, "   ",
avg)
            name = None
        else:
            tamname = len(name)
            string = ' '
            missingspaceamount = 15 - tamname
            spacequantity = missingspaceamount * string
            name15 = name + spacequantity
            print(''.join(name15),"    ", minimum, "   ", maximum, "   ", avg)
        checker = False
    return None
```

```
7- def request15daysdata(url, number, dados):

   This function is responsible for requesting the 15-Day data on the Web. The
difficulty found was to treat the corresponding URL, it has some endpoints that
must be concatenated to arrive at the original URL. The exception was treated when
there are no data available on the website.

    :param url {str}: Receives the default URL to be concatenated with the other
URL's
    :param number {str}: Identification number (ID) of the corresponding station
    :param dados {list}: The entire set of data read in load_stations ()

    :return:It has no return. This function serves as a bridge to be able to call
the function print (data) and print the results

    :error: Offline page errors were handled

    endpoint1 = "stations/"
    endpoint2 = "/W/measurements.json?start=P15D"
    try:
        response = urllib.request.urlopen(url + endpoint1 + number +
endpoint2)
        body = response.read().decode("utf-8")
        data15 = json.loads(body)
        list_dados = outputok(data15, number, dados)
        printdata(list_dados)
    except:
        list_dados = outputnone(number, dados)
        printdata(list_dados)
    return None
```

```
8- def request15dayssgraph(url, number, dados):

      This function is responsible for requesting data from the past 15 days of the
stations that will be plotted the graph. The difficulty found was to treat the
corresponding URL, it has some endpoints that must be concatenated to arrive at the
original URL. The exception was treated when there are no data available on the
website.

    :param url {str}: Receives the default URL to be concatenated with the other
URL's
    :param number {str}: Identification number (ID) of the corresponding station
    :param dados {list}: The entire set of data read in load_stations ()

    :return:It has no return. This function serves as a bridge to be able to call
the function printgraph(data) and to plot the results
```

```
    :error: Offline page errors were handled

    endpoint1 = "stations/"
    endpoint2 = "/W/measurements.json?start=P15D"
    try:
        response = urllib.request.urlopen(url + endpoint1 + number +
endpoint2)
        body = response.read().decode("utf-8")
        data15 = json.loads(body)
        list_dados = outputok(data15, number, dados)
        printgraph(list_dados)
    except:
        list_dados = outputnone(number, dados)
        printgraph(list_dados)
    return None


9- def printgraph(list):

    This function is responsible for plotting the graphs of all the stations that
were searched by the user. The difficulty found to perform this function was to try
to understand what was the mathematical form used to print the results.

    :param list {list}: It receives a list with the data of the station that is
wanted to plot the graph like this example: ('BONAFORTH', 233.0, 288.0,
253.6459930313589)

    :return: Because it is a printing function, it has no return

    :error: The difficulty found was in the adjustment of stations that exceed or
do not have registered water levels

    name = list[0]
    name10 = []
    minimum = float(list[1])
    maximum = float(list[2])
    avg = list[3]
    minimum = ("%4.2f" % minimum)
    maximum = ("%4.2f" % maximum)
    avg = ("%4.2f" % avg)
    checker = True
    while checker:
        if len(name) >= 10:
            for i in name:
                letra = i
                if len(name10) < 10:
                    name10.append(letra)
            if len(maximum) == 4:
                print(''.join(name10), '|----------------------------------------')
            else:
                equalsign = '='
                minussign = '-'
                barsign = '|'
                begin = math.ceil(float(minimum) /25)
                grafik = []
                for j in range(1, begin):
                    grafik.append(minussign)
                to = math.ceil((float(avg) - float(minimum)) /25) + begin
                for k in range(begin, to):
                    grafik.append(equalsign)
                grafik.append(barsign)
                to2 = math.ceil((float(maximum)-float(avg)) / 25) + to
                if (to2 < 42):
                    for l in range(to, to2):
                        grafik.append(equalsign)
                    for m in range(to2, 41):
                        grafik.append(minussign)
                else:
                    for l in range(to, 41):
                        grafik.append(equalsign)
                print(''.join(name10),''.join(grafik))
            name = None
        else:
            tamname = len(name)
            string = ' '
            missingspaceamount = 10 - tamname
            spacequantity = missingspaceamount * string
            name10 = name + spacequantity
            if len(maximum) == 4:
                print(''.join(name10), '|-----------------------------------------')
            else:
                equalsign = '='
                minussign = '-'
                barsign = '|'
                begin = math.ceil(float(minimum) / 25)
                grafik = []
                for j in range(1,begin):
                    grafik.append(minussign)
                to = math.ceil((float(avg) - float(minimum)) / 25) + begin
                for k in range(begin, to):
                    grafik.append(equalsign)
                grafik.append(barsign)
                to2 = math.ceil((float(maximum) - float(avg)) / 25) + to
                if (to2 < 42):
                    for l in range(to, to2):
                        grafik.append(equalsign)
                    for m in range(to2, 41):
                        grafik.append(minussign)
                else:
                    for l in range(to, 41):
                        grafik.append(equalsign)
                print(''.join(name10), ''.join(grafik))
        checker = False
    return None

10- def outputok(data15, number, dados):

     This function is responsible for handling the values of the web pages that
exist data to be calculated

    :param data15 {list}: All the data of the stations referring to last 15 days
    :param number{str}: It is the corresponding identifier of the station that will
be requested the data
    :param dados {list}: The entire dataset is passed as a parameter, because with
the number Identified will be extracted the name of the station

    :return: Returns the Name, Minimum value, Max value, and the previously
calculated arimetical mean, this data is passed to the function

    :error: No errors/difficult found

    allvalue = 0
    count = 0
    maxx = 0
    minn = 9999
    name = getname(number, dados)
    for i in data15:
        value = i.get('value')
        count += 1
        allvalue += value
        if value > maxx:
            maxx = value
        if value < minn:
            minn = value
    avg = allvalue / count
    return(name, minn, maxx, avg)



11- def outputnone(number, dados):

    This function is responsible for returning fixed values from the Pages of
stations that do not provide data.

    :param number {str}: The number is passed as a parameter because it will be
used to capture the station name using the get ()
    :param dados {list}: The entire dataset is passed as a parameter, because with
the number Identified will be extracted the name of the station

    :return:This function returns fixed values already stipulated by data
specifications that are not provided by Pregelonline

    :error: No errors found

    maxx = 0.00
    minn = 0.00
    avg = 0.00
    dados = dados
    number = number
    name = getname(number, dados)
    minn = ("%.2f" % minn)
    maxx = ("%.2f" % maxx)
    return(name, minn, maxx, avg)

12- def getname(number, dados):

    This function receives the station number, and the total data and returns the
station name

    :param number {str}: Number Identifier (ID)
    :param dados {list}: The entire data set is passed as param param, the
identifier number passed as parameter with the identifier (ID) of all the data if
they are the same will be extracted its name.

    :return:Returns the name of the station of the Identifier (ID) informed.

    :error: No errors found

    name = " "
    for i in dados:
        numero = i.get('number')
        if number == numero:
            name = i.get('shortname')
            name = name.upper()
    return(name)
```

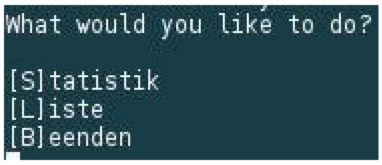
Algorithm

```
answer = True
while answer:
    print( Report: [S]tatistica, [L]ista ou  [F]im do Programa)
    IF answer.size == 1
        If letter == [S]tatistik
            print(Enter the station you are looking for)
            for each existing station

                all_station = name
            station_founded= Search_initial_letter (Informed_Letter, all_station)
            ID = return_identifiers_list(station_founded)
            IF id.size() != 0
                print( “Name“, “MIN“, “Maximo“, “AVG“)
            While counter < id.size()
                Request_15_days( url, id[counter], data)
        else if  letter == [L]ist
            print( List in alphabetical order of the waters stations)
            listed_station = list_station(data)
            for each station in listed_station
                print(station)
        else if letter == [G]grafik
            print( Please enter the initial characters of the Stations.)
            for each existing station
                all_station = name
            station_founded= Search_initial_letter (Informed_Letter, all_station)
            ID = return_identifiers_list(station_founded)
            if identifiers.size() !=0
                while cont < identifiers.size()
                    aux = identifiers(cont)
                    request15daysgraph(url,aux_identifiers,data)
                    cont +=1
        else if letter == [B]eenden (END)
            print(Thank you)
            answer = False
    else
        print (You must enter only the letters “S“, “L“ or “B“)
```

User Instruction Manual

Good morning user, the following will be presented a manual for better handling of the Software.

When you open the program, a Menu will appear as the following options:



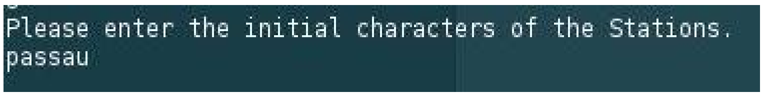
```
What would you like to do?

[S]tatistik
[L]iste
[B]eenden
```

Figure: 1

STATISTICAL MENU

The "Statistics" menu will analyze the water levels in the last 15 days. For example, type the letter "S" in order to activate this part of the Menu and later type the letters of the station that wants to view the statistics.

For example, for water stations in the city of Passau, enter "Passau"

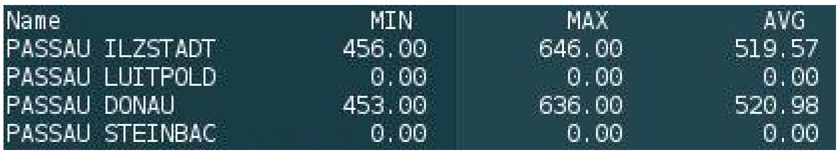


```
Please enter the initial characters of the Stations.
passau
```

Figure: 2

As can be seen there are 4 water stations in the city of Passau.

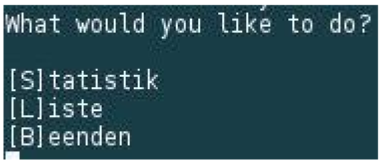
Subsequently, it is presented for each water station in a 15-day interval which is its lowest "Minimum" water level, the highest water level "Maximum" and also its AVG in the time period.



```
Name                    MIN          MAX          AVG
PASSAU ILZSTADT      456.00       646.00       519.57
PASSAU LUITPOLD        0.00         0.00         0.00
PASSAU DONAU         453.00       636.00       520.98
PASSAU STEINBAC        0.00         0.00         0.00
```

Figure: 3

The Menu in Figure 1 will then be displayed again. If you wish to make another query for another station, repeat this process to access the water level of the desired stations.



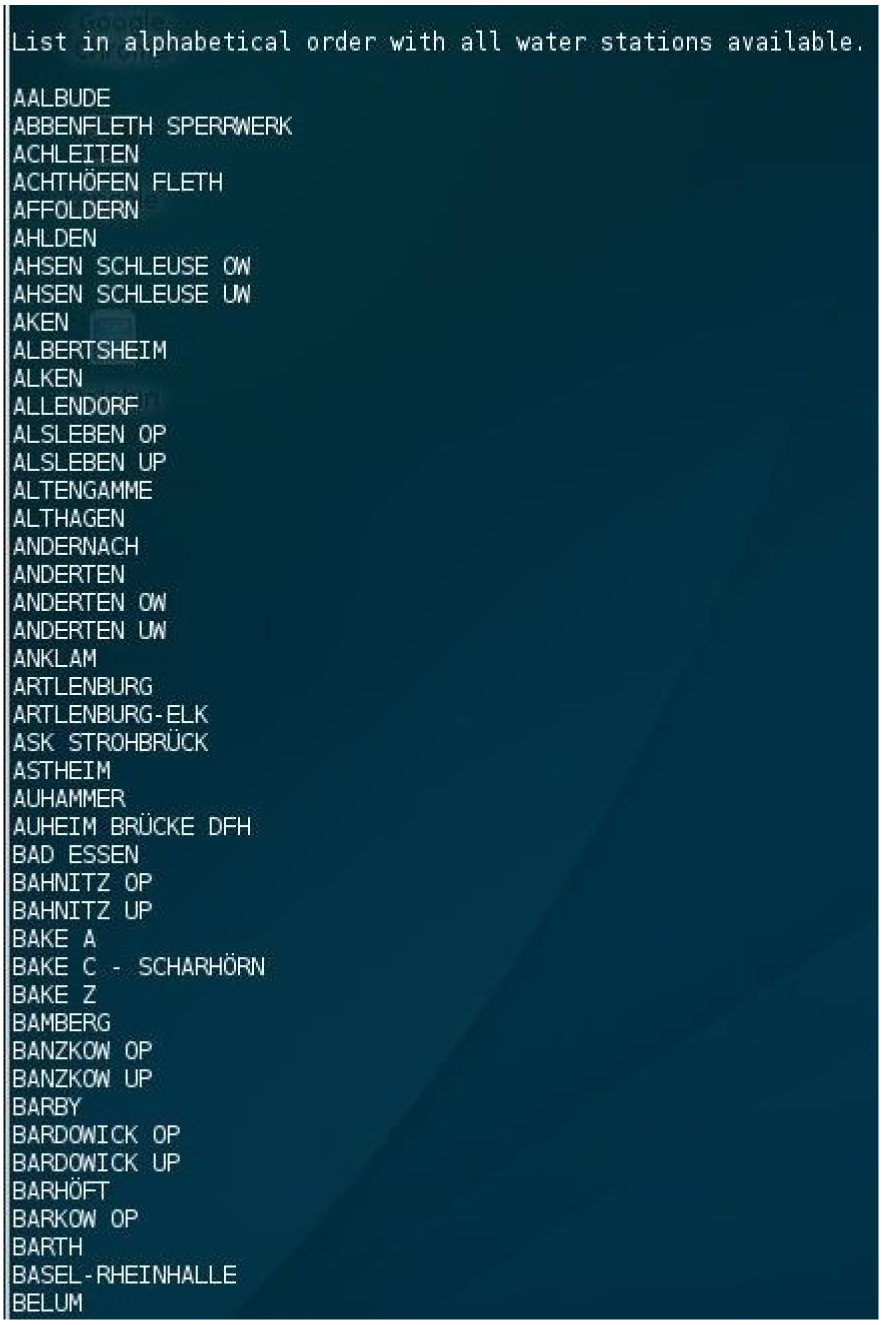
```
What would you like to do?

[S]tatistik
[L]iste
[B]eenden
```

Figure: 4

LIST MENU

If you want to know which water stations are registered, you can activate the "List" part of the Menu by entering the letter "L".

After entering "L" will be presented an alphabetical list with all the stations of water present in the site Pregelonline as shown below:



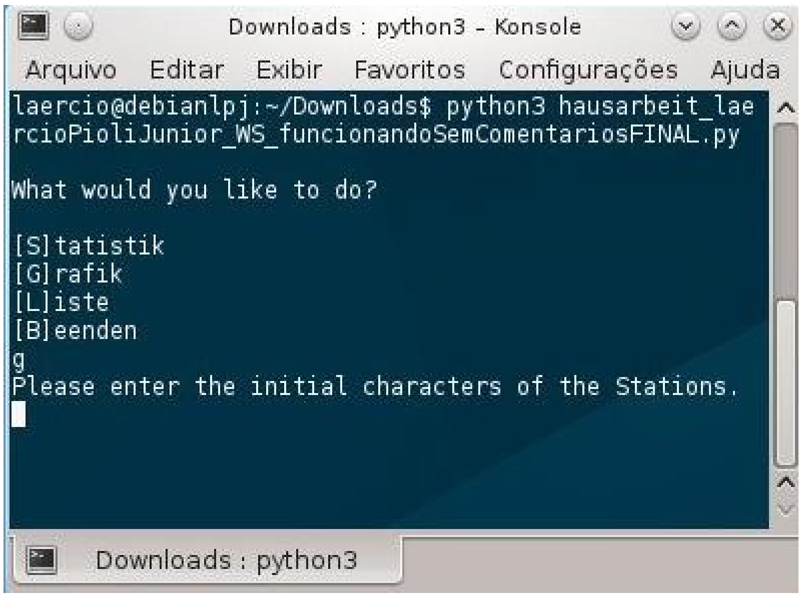
```
List in alphabetical order with all water stations available.

AALBUDE
ABBENFLETH SPERRWERK
ACHLEITEN
ACHTHÖFEN FLETH
AFFOLDERN
AHLDEN
AHSEN SCHLEUSE OW
AHSEN SCHLEUSE UW
AKEN
ALBERTSHEIM
ALKEN
ALLENDORF
ALSLEBEN OP
ALSLEBEN UP
ALTENGAMME
ALTHAGEN
ANDERNACH
ANDERTEN
ANDERTEN OW
ANDERTEN UW
ANKLAM
ARTLENBURG
ARTLENBURG-ELK
ASK STROHBRÜCK
ASTHEIM
AUHAMMER
AUHEIM BRÜCKE DFH
BAD ESSEN
BAHNITZ OP
BAHNITZ UP
BAKE A
BAKE C - SCHARHÖRN
BAKE Z
BAMBERG
BANZKOW OP
BANZKOW UP
BARBY
BARDOWICK OP
BARDOWICK UP
BARHÖFT
BARKOW OP
BARTH
BASEL-RHEINHALLE
BELUM
```

Figure: 5

GRAPHIC MENU

You can also access the "Graphic" menu. This menu allows the water level output of the searched stations to be represented by a bar graph.
For each registered station a graphic bar will be drawn. The bar graph is based on the statistics regarding the amount of water present in the station.
The total water level in each bar corresponds to 1000 cm. The values of Minimum, Maximum and aritimetic Average are represented in the graph through the following requirements.
* The "=" signs in the bars represent the water level for each station, the leftmost being the Minimum value and the rightmost the Maximum value.
* Each "-" drawn corresponds to a measure of 25 cm and the graph is assembled taking this information as true.
* The arithmetic average is represented graphically by the symbol "| ".

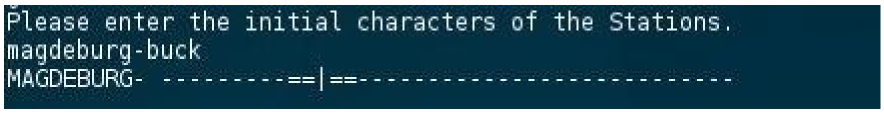
Here is an example containing to better understand how the graphics were assembled.
First select the Graph in the Start Menu and then enter the name of the station you want to query.


Downloads : python3 – Konsole
Arquivo Editar Exibir Favoritos Configurações Ajuda
Downloads : python3

```
laercio@debianlpj:~/Downloads$ python3 hausarbeit_lae
rcioPioliJunior_WS_funcionandoSemComentariosFINAL.py

What would you like to do?

[S]tatistik
[G]rafik
[L]iste
[B]eenden
g
Please enter the initial characters of the Stations.
```

Figure: 6

After inserting the water station, the graphic will be drawn on the screen as shown below:



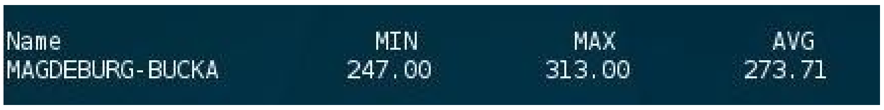
```
Please enter the initial characters of the Stations.
magdeburg-buck
MAGDEBURG- ---------==|==--------------------------
```

Figure:7

In this example the water station "MAGDEBURG-BUCKAU" was searched. Note that for return, the station's full name was not entered. The software is programmed to know that after entering the word "magdeburg-buck" the desired station would be "MAGDEBURG-BUCKAU".

But first let's check the Maximo, Minimo and Arithmetic Mean values in the "Statistic" menu for the desired station. Later we will search the same station in the "Chart" menu to analyze the gafico.

Figure: 8

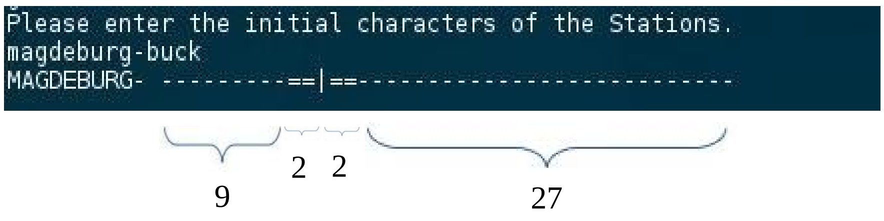


```
Name                    MIN        MAX        AVG
MAGDEBURG-BUCKA      247.00     313.00     273.71
```

247 / 25 = 9,88 273 / 25 = 10,92

313 / 25 = 12,52

Figure: 9



```
Please enter the initial characters of the Stations.
magdeburg-buck
MAGDEBURG- ---------==|==---------------------------
```

Each character of the bar varies from 25 in 25 cm of water. To carry out the analysis, one must take into account the variation of each value for its representative character in the graph. For example the first character varies from 0 → 25 cm of water and so on.

To represent the "MIN" in the graph, the minimum was divided by 25, but the result was 9.88 which suggests that there are 9 characters "-" before the character "=" representing the lowest water level

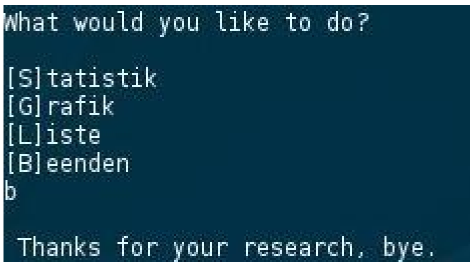
The maximum value was also divided by 25, thus (313/25 = 12.52). The graph varies in the largest integer value after the comma, so there must be signs of "=" from character 10 to character 13 in the graph.

The average represented by "| "Is the AVG value previously calculated, this calculation is done internally.

At the end of each query the figure: 1 will appear again showing you the Start Menu. If you want to make new queries to see more graphics, apply the steps again and the graphics will be displayed.

MENU EXIT

To Exit the program, simply press the "B" key in the Start Menu and the Program will be finished.



```
What would you like to do?

[S]tatistik
[G]rafik
[L]iste
[B]eenden
b

 Thanks for your research, bye.
```

Figura: 10

PROGRAMM

```
# -*- coding: utf-8 -*-

import json
import urllib.request
import math

def load_stations(url_param,endpoint_param):
    This function receives a URL and an Endpoint and loads the data (stations) into
a memory variable in Python format for later use in the program.


  :param url_param {str}: Base URL address for downloading all data
  :param endpoint_param {str}:Complementary address of the URL that characterizes
the water stations.
  :return:data: Returns a List with all the stations present on the Web. The entire
program will use the saved data present in this file.
  :error: Offline page errors were handled
    try:
        url = url_param
        endpoint = endpoint_param
        response = urllib.request.urlopen(url + endpoint)
        body = response.read().decode("utf-8")
        data = json.loads(body)
    except:
        print("Website is offline, please try again later")
    return data



def stationen_auflisten(dados):

     This function returns a dictionary with all water stations, where the key is
its alphabetical position and value is the name of the station.

    :param dados {list}: Receives a list of dictionaries, where each dictionary is
a water station with several attributes (uuid, number, shortname, longname, km,
agency, longitude, latitude, water

    :return:Returns a dictionary in alphabetical order, which will aid the
presentation of item 2 [L] iste

    :error: No errors/difficult found

    data = dados
    new_dict = {item['longname']: item for item in data}
    allstations = {}
    count = 0
    for i in sorted(new_dict):
        allstations[count] = i
        count += 1
    return allstations


def searchinitialletter (initialletter, list_cidades):

    This function searches for the letters entered by the user in the list
of stations passed by parameter to see if that station exists.

    :param initialletter {str}: Letter entered by the user
    :param list_cidades {list}: List with all the stations that are in the data
document
```

```
    :return: Returns a list with all the stations with the name that the user
entered if it exists.

    :error: No errors/difficult found

    i = 0
    aux_stationfound = []
    while i < len(list_cidades):
        if initialletter in list_cidades[i]:
            aux_stationfound.append(list_cidades[i])
        i += 1
    if len(aux_stationfound) == 0:
        print("This station could not be identified, please try again")
    return aux_stationfound


def returnnumberid (namecidade,dados):

     This function receives as a parameter a name of a station and the total data
set and returns the identified station.

    :param namecidade {str}: The name of a particular station
    :param dados {list}: Total set of data read with the load_stations function

    :return:Returns a String with the corresponding station identifier.

    :error: No errors/difficult found

    number = None
    for i in dados:
        cidade_dados = i.get('longname')
        if namecidade == cidade_dados:
            number = i.get('number')
        else:
            False
    return number

def returnlistid(list_stationfound):

     This function receives as a parameter a list with the stations that were found
when the user searched the [E] tatistic (statistic).

    :param list_stationfound {list}:Receives the name of the stations found in the
data.

    :return: Returns a list with all the identifiers of the stations found

    :error: No errors/difficult found

    j = 0
    list_id = []
    while j < len(list_stationfound):
        id = returnnumberid(list_stationfound[j], dados)
        list_id.append(id)
        j += 1
    return list_id


def printdata(list):

     This function receives a list like this ['PASSAU ILZSTADT', 456.0, 608.0,
501.8209169054441] and applies the formatting required to be able to print the
data.

    :param list {list}:This list contains the values (station name, Minimo, Maximo
and AVG) that have already been calculated

    :return: No return
    :error: No errors/difficult found

    name = list[0]
    name15 = []
    minimum = float(list[1])
    maximum = float(list[2])
    avg = list[3]
    avg = ("%10.2f" % avg)
    minimum = ("%10.2f" % minimum)
    maximum = ("%10.2f" % maximum)
    checker = True
    while checker:
        if len(name) > 15:
            for i in name:
                letra = i
                if len(name15) < 15:
                    name15.append(letra)
            print(''.join(name15), "    ", minimum, "   ", maximum, "   ",
avg)
            name = None
        else:
            tamname = len(name)
            string = ' '
            missingspaceamount = 15 - tamname
            spacequantity = missingspaceamount * string
            name15 = name + spacequantity
            print(''.join(name15),"    ", minimum, "   ", maximum, "   ", avg)
        checker = False
    return None

def request15daysdata(url, number, dados):

    This function is responsible for requesting the 15-Day data on the Web. The
difficulty found was to treat the corresponding URL, it has some endpoints that
must be concatenated to arrive at the original URL. The exception was treated when
there are no data available on the website.

    :param url {str}: Receives the default URL to be concatenated with the other
URL's
    :param number {str}: Identification number (ID) of the corresponding station
    :param dados {list}: The entire set of data read in load_stations ()

    :return:It has no return. This function serves as a bridge to be able to call
the function print (data) and print the results

    :error: Offline page errors were handled

    endpoint1 = "stations/"
    endpoint2 = "/W/measurements.json?start=P15D"
    try:
        response = urllib.request.urlopen(url + endpoint1 + number +
endpoint2)
        body = response.read().decode("utf-8")
        data15 = json.loads(body)
        list_dados = outputok(data15, number, dados)
        printdata(list_dados)
    except:
        list_dados = outputnone(number, dados)
        printdata(list_dados)
    return None



def request15dayssgraph(url, number, dados):

       This function is responsible for requesting data from the past 15 days of the
stations that will be plotted the graph. The difficulty found was to treat the
```

```
corresponding URL, it has some endpoints that must be concatenated to arrive at the
original URL. The exception was treated when there are no data available on the
website.

    :param url {str}: Receives the default URL to be concatenated with the other
URL's
    :param number {str}: Identification number (ID) of the corresponding station
    :param dados {list}: The entire set of data read in load_stations ()

    :return:It has no return. This function serves as a bridge to be able to call
the function printgraph(data) and to plot the results

    :error: Offline page errors were handled

    endpoint1 = "stations/"
    endpoint2 = "/W/measurements.json?start=P15D"
    try:
        response = urllib.request.urlopen(url + endpoint1 + number +
endpoint2)
        body = response.read().decode("utf-8")
        data15 = json.loads(body)
        list_dados = outputok(data15, number, dados)
        printgraph(list_dados)
    except:
        list_dados = outputnone(number, dados)
        printgraph(list_dados)
    return None


def printgraph(list):

    This function is responsible for plotting the graphs of all the stations that
were searched by the user. The difficulty found to perform this function was to try
to understand what was the mathematical form used to print the results.

    :param list {list}: It receives a list with the data of the station that is
wanted to plot the graph like this example: ('BONAFORTH', 233.0, 288.0,
253.6459930313589)

    :return: Because it is a printing function, it has no return

    :error: The difficulty found was in the adjustment of stations that exceed or
do not have registered water levels

    name = list[0]
    name10 = []
    minimum = float(list[1])
    maximum = float(list[2])
    avg = list[3]
    minimum = ("%4.2f" % minimum)
    maximum = ("%4.2f" % maximum)
    avg = ("%4.2f" % avg)
    checker = True
    while checker:
        if len(name) >= 10:
            for i in name:
                letra = i
                if len(name10) < 10:
                    name10.append(letra)
            if len(maximum) == 4:
                print(''.join(name10), '|-------------------------------------')
            else:
                equalsign = '='
                minussign = '-'
                barsign = '|'
                begin = math.ceil(float(minimum) /25)
                grafik = []
                for j in range(1, begin):
                    grafik.append(minussign)
                to = math.ceil((float(avg) - float(minimum)) /25) + begin
                for k in range(begin, to):
                    grafik.append(equalsign)
                grafik.append(barsign)
                to2 = math.ceil((float(maximum)-float(avg)) / 25) + to
                if (to2 < 42):
                    for l in range(to, to2):
                        grafik.append(equalsign)
                    for m in range(to2, 41):
                        grafik.append(minussign)
                else:
                    for l in range(to, 41):
                        grafik.append(equalsign)
                print(''.join(name10),''.join(grafik))
            name = None
        else:
            tamname = len(name)
            string = ' '
            missingspaceamount = 10 - tamname
            spacequantity = missingspaceamount * string
            name10 = name + spacequantity
            if len(maximum) == 4:
                print(''.join(name10), '|-----------------------------------------')
            else:
                equalsign = '='
                minussign = '-'
                barsign = '|'
                begin = math.ceil(float(minimum) / 25)
                grafik = []
                for j in range(1,begin):
                    grafik.append(minussign)
                to = math.ceil((float(avg) - float(minimum)) / 25) + begin
                for k in range(begin, to):
                    grafik.append(equalsign)
                grafik.append(barsign)
                to2 = math.ceil((float(maximum) - float(avg)) / 25) + to
                if (to2 < 42):
                    for l in range(to, to2):
                        grafik.append(equalsign)
                    for m in range(to2, 41):
                        grafik.append(minussign)
                else:
                    for l in range(to, 41):
                        grafik.append(equalsign)
                print(''.join(name10), ''.join(grafik))
        checker = False
    return None

def outputok(data15, number, dados):

      This function is responsible for handling the values of the web pages that
exist data to be calculated

    :param data15 {list}: All the data of the stations referring to last 15 days
    :param number{str}: It is the corresponding identifier of the station that will
be requested the data
    :param dados {list}: The entire dataset is passed as a parameter, because with
the number Identified will be extracted the name of the station

    :return: Returns the Name, Minimum value, Max value, and the previously
calculated arimetical mean, this data is passed to the function

    :error: No errors/difficult found

    allvalue = 0
    count = 0
```

```python
    maxx = 0
    minn = 9999
    name = getname(number, dados)
    for i in data15:
        value = i.get('value')
        count += 1
        allvalue += value
        if value > maxx:
            maxx = value
        if value < minn:
            minn = value
    avg = allvalue / count
    return(name, minn, maxx, avg)



def outputnone(number, dados):

    This function is responsible for returning fixed values from the Pages of
stations that do not provide data.

    :param number {str}: The number is passed as a parameter because it will be
used to capture the station name using the get ()
    :param dados {list}: The entire dataset is passed as a parameter, because with
the number Identified will be extracted the name of the station

    :return:This function returns fixed values already stipulated by data
specifications that are not provided by Pregelonline

    :error: No errors found

    maxx = 0.00
    minn = 0.00
    avg = 0.00
    dados = dados
    number = number
    name = getname(number, dados)
    minn = ("%.2f" % minn)
    maxx = ("%.2f" % maxx)
    return(name, minn, maxx, avg)

def getname(number, dados):

    This function receives the station number, and the total data and returns the
station name

    :param number {str}: Number Identifier (ID)
    :param dados {list}: The entire data set is passed as param param, the
identifier number passed as parameter with the identifier (ID) of all the data if
they are the same will be extracted its name.

    :return:Returns the name of the station of the Identifier (ID) informed.

    :error: No errors found

    name = " "
    for i in dados:
        numero = i.get('number')
        if number == numero:
            name = i.get('shortname')
            name = name.upper()
    return(name)
```

```
#----------The data is downloaded and saved in the variable data---------#

'''         A fixed URL and a fixed endpoint are used to extract the data
'''
url =  'http://www.pegelonline.wsv.de/webservices/rest-api/v2/'
endpoint = 'stations.json'
dados = load_stations(url, endpoint)
stationen_aufgelisted = {}
answer=True
while answer:
    answermenu = input("\nWhat would you like to
do?\n \n[S]tatistik\n[G]rafik\n[L]iste\n[B]eenden \n")
    if len(answermenu) == 1:
        if (ord(answermenu)==83) or (ord(answermenu) == 115):
            initialletter = " "
            initialletter = input("Please enter the initial characters of
the Stations.\n")
            initialletter = initialletter.upper()
            allstation = []
            stationfound = []
            identifiers = []
            for d in dados:
                aux = d.get('longname')
                allstation.append(aux)
            stationfound = searchinitialletter(initialletter,allstation)
            identifiers = returnlistid(stationfound)
            if len(identifiers) != 0:
                print("\nName", "                    ", "MIN", "         ",
"MAX", "          ", "AVG")
            cont = 0
            while cont < len(identifiers):
                aux_identifiers = identifiers[cont]
                request15daysdata(url, aux_identifiers, dados)
                cont += 1
        elif (ord(answermenu)==76) or (ord(answermenu) == 108):
          print("\nList in alphabetical order with all water stations
available.\n")
          stationen_aufgelisted = stationen_auflisten(dados)
          for k, v in stationen_aufgelisted.items():
              print(v)
        elif (ord(answermenu)==71) or (ord(answermenu) == 103):
            initialletter = input("Please enter the initial characters of
the Stations.\n")
            initialletter = initialletter.upper()
            allstation = []
            stationfound = []
            identifiers = []
            for d in dados:
                aux = d.get('longname')
                allstation.append(aux)
            stationfound = searchinitialletter(initialletter,allstation)
            identifiers = returnlistid(stationfound)
            if len(identifiers) != 0:
                cont = 0
                while cont < len(identifiers):
                    aux_identifiers = identifiers[cont]
                    request15dayssgraph(url, aux_identifiers, dados)
                    cont += 1
        elif (ord(answermenu)==66) or (ord(answermenu) == 98):
          answer = False
          print("\n Thanks for your research, bye.")
    else:
        print("You must enter only the letters, 'S', 'L', 'G' or 'B'")
```