

02 files, loops, conditions

0.1 First loop

Let's repeat, let's write the program again, giving it the temperature in Celsius degrees and letting us out the temperature in Fahrenheit.

```
[1]: vnos = input("Temperatura: ")

temp_c = int(vnos)
temp_f = temp_c * 9 / 5 + 32
print(temp_c, "C je", temp_f, "F")
```

Temperatura: 25

25 C je 77.0 F

Vnos = entry

We divided it into two parts and put a blank line between them. The first is short - just a line to provide the data for the second part, which calculates and prints.

Now let's set our sights higher: in the file "temperature.txt" we have the predicted temperature data in Radovljica for the week of 2 to 6 October 2023. The contents of the file are as follows:

24
18
15
16
18

Our task is to run a programme for each of these temperatures. It's as if we were entering them by hand, only it will read them from a file. So, we would like to have a printout like this:

24 C je 75.2 F

18 C je 64.4 F

15 C je 59.0 F

16 C je 60.8 F

18 C je 64.4 F

We need to modify the program to repeat the second part for each temperature in the file. This is done by:

```
[2]: for vnos in open("temperature.txt"):
    temp_c = int(vnos)
    temp_f = temp_c * 9 / 5 + 32
    print(temp_c, "C je", temp_f, "F")
```

24 C je 75.2 F

18 C je 64.4 F

15 C je 59.0 F
16 C je 60.8 F
18 C je 64.4 F
13 C je 55.4 F
15 C je 59.0 F
15 C je 59.0 F
10 C je 50.0 F
12 C je 53.6 F
20 C je 68.0 F
10 C je 50.0 F
15 C je 59.0 F

The last three lines are as before, but the first is new. *open* is a function. (How do we know? By calling it. Just as we wrote the braces for *input*, *int* and *print*, here we write them for *open*, so *open* is a function. If a name is followed by parentheses, it's always means a function call.) As an argument to it, we obviously give it a filename. What does it do, what does it return?

The *open* function "opens" the file. If it were Word, Excel or, uh, VLC, for example, this would mean, it shows text, opens a spreadsheet, starts playing a movie. But in programming languages, the functions *open* and similar functions (which may have different names in different programming languages) simply search (or I know, it's still a bit abstract, but it will stay that way. The *open* function returns something, some thing, some object that represents the file. It returns a "box", which is not, as we are used to now, *int* or *float* or *str* but something fourth, a file. (Anyone who expects to be told the name of a data type, for example *file*, is waiting in vain. The name of the data type is strange, and it depends on the type of file. Besides, this name is never we don't need it. So never that your lecturer for Programming 1 doesn't even know how to use this type - or variants of these types - are really called.)

Here, we learn about the operations that can be performed on files. While strings and numbers can be added, multiplied, and divided, files work differently. The only operation that can be done with files is querying them, which returns consecutive lines from the file. To query a file in Python, we use the syntax `for input in open("temperature.txt"):`. The first line of the file is requested and its contents are stored in a variable called "input". The lines of code that are written below the "for" statement and indented will then be executed. After executing these lines, the next line from the file will be requested, assigned to a variable called "entry", and the lines will be executed again. This process continues until the end of the file is reached.

In computer science, a loop is a repetitive structure that allows for the execution of a specific set of instructions multiple times. In this context, the term "loop" refers to a repetition. A loop consists of a header, which includes an input line and a condition that determines when the loop will end. The body of the loop contains the instructions that are repeated. Python has two types of loops, one of which is the for loop. The header of a for loop includes a variable name and an iterable object. The body of the loop follows on the next lines and must be indented. If necessary, additional code can be written after the loop without indentation. Let's see:

```
[3]: for vnos in open("temperature.txt"):
    temp_c = int(vnos)
    temp_f = temp_c * 9 / 5 + 32
    print(temp_c, "C je", temp_f, "F")
print("Da, dragi moji, tako toplo bo prihodnji teden v Radoljci!")
```

= "Yes, my dears, it's going to be so warm next week in Radoljca!"

24 C je 75.2 F

18 C je 64.4 F

15 C je 59.0 F

16 C je 60.8 F

18 C je 64.4 F

13 C je 55.4 F

15 C je 59.0 F

15 C je 59.0 F

10 C je 50.0 F

12 C je 53.6 F

20 C je 68.0 F

10 C je 50.0 F

15 C je 59.0 F

Da, dragi moji, tako toplo bo prihodnji teden v Radoljci!

```
[4]: for vnos in open("temperature.txt"):
      temp_c = int(vnos)
      temp_f = temp_c * 9 / 5 + 32
      print(temp_c, "C je", temp_f, "F")
      print("Da, dragi moji, tako toplo bo prihodnji teden v Radoljci")
```

24 C je 75.2 F

18 C je 64.4 F

15 C je 59.0 F

16 C je 60.8 F

18 C je 64.4 F

13 C je 55.4 F

15 C je 59.0 F

15 C je 59.0 F

10 C je 50.0 F

12 C je 53.6 F

20 C je 68.0 F

10 C je 50.0 F

15 C je 59.0 F

Da, dragi moji, tako toplo bo prihodnji teden v Radoljci!

Here: the last print-command is in the zanka/loop as well:

```
for vnos in open("temperature.txt"):
    temp_c = int(vnos)
    temp_f = temp_c * 9 / 5 + 32
    print(temp_c, "C je", temp_f, "F")
    print("Da, dragi moji, tako toplo bo prihodnji teden v Radoljci")
```

24 C je 75.2 F

Da, dragi moji, tako toplo bo prihodnji teden v Radoljci

18 C je 64.4 F

Da, dragi moji, tako toplo bo prihodnji teden v Radoljci

15 C je 59.0 F

Da, dragi moji, tako toplo bo prihodnji teden v Radoljci

...

For delays in Python it is recommended to use spaces. Furthermore, in any case, the indentation must be consistent within a block.

0.2 Account loop

For this purpose, we will be calculating Celsius to Fahrenheit. The idea is: in each step of the loop, we need the variable sum of the variable, which is obtained by adding the value from the line to the previous value of sum file. Python only complains that it doesn't know the variable named sum. This happens because in the very first round we try to add to sum before that variable exists. This is easy to solve: before the loop, we set it to 0. Then we'll be adding nicely.

```
vsota = 0
for vnos in open("temperature.txt"):
    vsota = vsota + vnos
```

But we know this. sum is int, input is text, p. input will need to be converted to a number. (By the way, let's rename it to something more clever, say temperature or line.)

```
: vsota = 0
  for vrstica in open("temperature.txt"):
      vsota = vsota + int(vrstica)
  print(vsota)
```

201

We, of course, need an average. If we know that a file contains exactly 5 temperatures, we can of course print out sum / 5. But if we want to be more general, we'll keep counting as we add up days.

```
vsota = 0
dni = 0
for vrstica in open("temperature.txt"):
    vsota = vsota + int(vrstica)
    dni += 1
print(vsota / dni)
```

15.461538461538462

0.3 Conditional sentences

let's list all temperatures greater than 17. The task we have set ourselves is to print out only those of these numbers that are greater than 17. should therefore not always happen, but only if temp > 17.

```
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp > 17:
        print(temp)
```

24

18

18

20

Just as for asks Python to repeat something, if asks Python to do something only if it is a certain condition is met. We call for (and later while) a loop. Some people call if a loop too, but only beginner students. An if (together with what follows it) is a conditional statement. Like a for line, an if line must be terminated by a colon and followed by one or more or more indented lines. Since the if itself is indented by four spaces (because it is inside the for), it will print, which is inside if, is indented by eight spaces.

An interlude for socially- or even linguistically-oriented readers. When I wrote the above sentence, I initially forgot the last comma, the one before "offset". These spacing and indentation are practically the same thing as there are subordinate clauses in a language, which can also be subordinate, like this one, to other to other subordinate clauses, isn't that so? These lags and digressions are practically the same thing, as subordinate clauses are in language, which can also be subordinate, like, say, this one, other subordinate clauses, isn't that so?

In Slovene, the beginning and the end of a subordinate clause are indicated by a comma (which, unfortunately, doesn't tell us whether we're going inside the subordinate clause). or out), but in Python we do it with trailing spaces and, for clarity, a colon at the trailing space.)

And if you just want to count how many warm days there will be? In the case of this data, there are three; so we just want to 3, not individual temperatures. In this case, we replace the print with a count.

```
toplih = 0
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp > 17:
        toplih = toplih + 1
print("Toplih dni bo", toplih)
```

Toplih dni bo 4

That doesn't sound very nice. Let's set a different goal here:

We want to say that there will be 3 out of 5 warm days.

To do this, we need to count all the days in addition to the warm days. We already know how to do that.

```
toplih = 0
dni = 0
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp > 17:
```

```
        toplih = toplih + 1
        dni = dni + 1
print("Toplih dni bo", toplih, "od", dni)
```

Toplih dni bo 4 od 13

Do we see any deviations? The line warm = warm + 1 is inside the if, so it is executed only for those lines files that contain temperatures greater than 17 The line days = days + 1 is not inside the if, but it is inside for. Therefore, it is executed for every line of the file. If it were to be moved further, out of the loop, it would be executed only once.

```
toplih = 0
dni = 0
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp > 17:
        toplih = toplih + 1
dni = dni + 1
print("Toplih dni bo", toplih, "od", dni)
```

Toplih dni bo 4 od 1

0.4 Otherwise

Next task: someone wants to know if the temperature will be at least 20 degrees sometime next week.

So we would like a program that says either "Yes, there will be a warm day next week". or "No, next week will be one of gloom."

We will soon learn a more proper way of solving this problem, but we are up to it even with what we know so far. Let's simply count how many days the temperature will be at least 20 degrees and then check, if there are more than 0 days.

```
nad_20 = 0
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp > 19:
        nad_20 = nad_20 + 1

if nad_20 > 0:
    print("Da, v prihodnjem tednu bo tudi kak topel dan.")
```

Da, v prihodnjem tednu bo tudi kak topel dan.

The programme is working, but it will only be running for about a week, maybe two. That is, until the ARSO reports that "temperatures will be relatively high for this time of year". As soon as they drop below 20, the programme will not work. output anything.

```
nad_20 = 0
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp > 19:
        nad_20 = nad_20 + 1

if nad_20 > 0:
    print("Da, v prihodnjem tednu bo tudi kak topel dan.")
if nad_20 < 1:
    print("Ne, prihodnji teden bo ena žalost.")
```

Da, v prihodnjem tednu bo tudi kak topel dan.

Conditional sentences are sometimes used to tell something to happen only in a certain case. Often, we use them we're setting up two scenarios: if the condition is true, do one thing, otherwise do something else. This is already the case: if a day is above_20, it should say there will be warm days, otherwise it should say there will not be warm days. In all normal programming languages, if is allowed to be followed by else.

```
nad_20 = 0
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp > 19:
        nad_20 = nad_20 + 1

if nad_20 > 0:
    print("Da, v prihodnjem tednu bo tudi kak topel dan.")
else:
    print("Ne, prihodnji teden bo ena žalost.")
```

Da, v prihodnjem tednu bo tudi kak topel dan.

```

nad_20 = 0
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp > 19:
        nad_20 = nad_20 + 1

if nad_20 > 0:
    print("Da, v prihodnjem tednu bo tudi kak topel dan.")
    print("Le glej, da jih boš izkoristil!")
else:
    print("Ne, prihodnji teden bo ena žalost.")
    print("Zima bo, kaj hočemo.")
    print("Še malo, pa se bomo sankali.")
print("Ampak vse to je, pazi, samo napoved.")

```

Da, v prihodnjem tednu bo tudi kak topel dan.
 Le glej, da jih boš izkoristil!
 Ampak vse to je, pazi, samo napoved.

Unlike if, else has no condition. The condition is already written above, in the if. The lines that follow else will be executed when the condition is false.

If we have more than two nice days, we will say that the week will be very nice and it is a pity that you cannot take leave. Otherwise, if there is only one nice day, let us warn that there will be only one nice day and that it is worth taking advantage of it. Otherwise we groan.

```

nad_20 = 0
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp > 19:
        nad_20 = nad_20 + 1

if nad_20 > 1:
    print("Kako lep teden, vsaj dva dneva bosta topla!")
    print("Kakšna škoda, da moramo hoditi na predavanja!")
if nad_20 > 0:
    print("En sam topel dan!")
    print("Glej, da ga izkoristiš!")
else:
    print("Prihodnji teden bo ena žalost.")
    print("Zima bo, kaj hočemo.")

```

Kako lep teden, vsaj dva dneva bosta topla!
 Kakšna škoda, da moramo hoditi na predavanja!
 En sam topel dan!
 Glej, da ga izkoristiš!

It works, but do we see the problem? Let's temporarily lower the temperature, instead of > 19, let's check > 10.

```

nad_20 = 0
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp > 10:
        nad_20 = nad_20 + 1

if nad_20 > 1:
    print("Kako lep teden, vsaj dva dneva bosta topla!")
    print("Kakšna škoda, da moramo hoditi na predavanja!")
if nad_20 > 0:
    print("En sam topel dan!")
    print("Glej, da ga izkoristiš!")
else:
    print("Prihodnji teden bo ena žalost.")
    print("Zima bo, kaj hočemo.")

```

```

Kako lep teden, vsaj dva dneva bosta topla!
Kakšna škoda, da moramo hoditi na predavanja!
En sam topel dan!
Glej, da ga izkoristiš!

```

The second if checks if we have a warm day. But it should only do this if it hasn't just been printed out before, that there will be (at least) two. The correct (er, more correct) way is:

```

: nad_20 = 0
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp > 10:
        nad_20 = nad_20 + 1

if nad_20 > 1:
    print("Kako lep teden, vsaj dva dneva bosta topla!")
    print("Kakšna škoda, da moramo hoditi na predavanja!")
else:
    if nad_20 > 0:
        print("En sam topel dan!")
        print("Glej, da ga izkoristiš!")
    else:
        print("Prihodnji teden bo ena žalost.")
        print("Zima bo, kaj hočemo.")

```

```

Kako lep teden, vsaj dva dneva bosta topla!
Kakšna škoda, da moramo hoditi na predavanja!

```

Instead of all these if,else's we can just write "elif":

```

nad_20 = 0
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp > 10:
        nad_20 = nad_20 + 1

if nad_20 > 1:
    print("Kako lep teden, vsaj dva dneva bosta topla!")
    print("Kakšna škoda, da moramo hoditi na predavanja!")

```



```

elif nad_20 > 0:
    print("En sam topel dan!")
    print("Glej, da ga izkoristiš!")
else:
    print("Prihodnji teden bo ena žalost.")
    print("Zima bo, kaj hočemo.")

```

Kako lep teden, vsaj dva dneva bosta topla!
 Kakšna škoda, da moramo hoditi na predavanja!

To check that it is working, raise the desired temperature back to 20 degrees.

```

nad_20 = 0
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp > 19:
        nad_20 = nad_20 + 1

if nad_20 > 1:
    print("Kako lep teden, vsaj dva dneva bosta topla!")
    print("Kakšna škoda, da moramo hoditi na predavanja!")
elif nad_20 > 0:
    print("En sam topel dan!")
    print("Glej, da ga izkoristiš!")
else:
    print("Prihodnji teden bo ena žalost.")
    print("Zima bo, kaj hočemo.")

```

Kako lep teden, vsaj dva dneva bosta topla!
 Kakšna škoda, da moramo hoditi na predavanja!

0.5 Other comparison operators

So we have the <, <=, > and >= operators to compare.

```

nad_20 = 0
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp >= 20:
        nad_20 = nad_20 + 1

if nad_20 >= 2:
    print("Kako lep teden, vsaj dva dneva bosta topla!")
    print("Kakšna škoda, da moramo hoditi na predavanja!")
elif nad_20 >= 0:
    print("En sam topel dan!")
    print("Glej, da ga izkoristiš!")
else:
    print("Prihodnji teden bo ena žalost.")
    print("Zima bo, kaj hočemo.")

```

Kako lep teden, vsaj dva dneva bosta topla!
 Kakšna škoda, da moramo hoditi na predavanja!

That is better, but in the second condition we would actually like to ask whether we have one such day. Maybe So?:

```
nad_20 = 0
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp >= 20:
        nad_20 = nad_20 + 1

if nad_20 >= 2:
    print("Kako lep teden, vsaj dva dneva bosta topla!")
    print("Kakšna škoda, da moramo hoditi na predavanja!")
elif nad_20 = 1:
    print("En sam topel dan!")
    print("Glej, da ga izkoristiš!")
else:
    print("Prihodnji teden bo ena žalost.")
    print("Zima bo, kaj hočemo.")
```

Cell In[31], line 10

```
elif nad_20 = 1:
```

SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?

Python (and most others) uses a double equals sign for comparison!

```
nad_20 = 0
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp >= 20:
        nad_20 = nad_20 + 1

if nad_20 >= 2:
    print("Kako lep teden, vsaj dva dneva bosta topla!")
    print("Kakšna škoda, da moramo hoditi na predavanja!")
elif nad_20 == 1:
    print("En sam topel dan!")
    print("Glej, da ga izkoristiš!")
else:
    print("Prihodnji teden bo ena žalost.")
    print("Zima bo, kaj hočemo.")
```

Kako lep teden, vsaj dva dneva bosta topla!
Kakšna škoda, da moramo hoditi na predavanja!

0.6 Comparing strings

With the <, <=, >, >=, == and != operators, we can also compare strings. It compares strings in much the same way as one would expect: alphabetically. But only quite because the capital letters come before the lower case letters in the alphabet. Because the numbers come before the letters. Because the quotation mark comes before the number. The curly brackets are after the letters, and the square brackets are "alphabetically" between the capital and the lower case letters. The alphabets are ... somewhere, but definitely after the letters.

0.7 The classics: maximum and minimum

Let's write a program that prints the maximum temperature. Imagine someone reading these numbers to us. What would you repeat in your head? The maximum that we've heard so far. Each successive number we hear, we compare it to the highest ever and if it is higher than the highest, we remember it (and forget the previous one). Sort of:

```
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp > najvisja_doslej:
        najvisja_doslej = temp

print(najvisja_doslej)
```

```
-----
NameError                                Traceback (most recent call last)
Cell In[33], line 3
```

```
1 for vrstica in open("temperature.txt"):
2     temp = int(vrstica)
----> 3     if temp > najvisja_doslej:
4         najvisja_doslej = temp
6 print(najvisja_doslej)

NameError: name 'najvisja_doslej' is not defined
```

What has happened to us has happened: we cannot use the variable `najvisja_doslej` ("highest_so far") inside a loop unless we set it before the loop. We will also rename it to "highest"/"najvisja". Hm, what value are we going to give it before the loop? For now, with our knowledge, it should be something like absurdly low, so the next temperature will be higher. At least as far as temperatures are concerned, we are safe: physicists teach us that the temperature cannot be lower than 0 K, so -274 will be a safe initial the safe value.

```
najvisja = -274
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp > najvisja:
        najvisja = temp

print(najvisja)
```

24

The minimum predicted temperature is calculated in the same way, but in reverse. However, the initial value can be millions. We will learn to do it more correctly some other time. So, to avoid repeating essentially the same program, let's try something more difficult: in addition to the lowest temperature, we are also interested in the sequence number of the day with that temperature. Again, think about how we would do this if someone was telling us to read the numbers. In addition to the lowest number, we would have to remember two more numbers: the sequential number of the current day (we already know this: remember how we counted the days when we worked out how many days out of which it would be warm), and we need to remember at the same time as the lowest number, we must remember the number of the current day at that time.

```
najnizja = 1000000
dan = 0
for vrstica in open("temperature.txt"):
    dan = dan + 1 # štejemo dneve
    temp = int(vrstica)
    if temp < najnizja: # naleteli smo na nov rekord
        najnizja = temp # zapomnimo si to, rekordno temperaturo
        najhladnejsi_dan = dan # in to številko dneva
print("Najnižja temperatura bo", najnizja, "C in bo nastopila na dan",
      najhladnejsi_dan)
```

```
Najnižja temperatura bo 10 C in bo nastopila na dan 9
```

0.8 Observing sequences

Three old fortune-tellers put their heads together and predicted the December temperatures. With the help of their grandson they typed them into a file called "december.txt":

-1
-6
-8
25
-6
-8
-12
-15
67
-20
23

New exercise: write a program that tells what the (expected) longest sequence of days is when the temperature is below zero. We will have to remember two things: how long the current sequence of days with negative temperature is and how long the longest sequence is. Every the loop gets a number and checks whether its negative or not. If it is not, then we “reset” the length of the sequence to 0. If it is negative, we add 1 to the length of the sequence and check whether we have accidentally obtained a sequence longer than the longest.

```
dolzina = 0
naj_dolzina = 0
for vrstica in open("december.txt"):
    temp_c = int(vrstica)
    if temp_c < 0:
        dolzina = dolzina + 1
        if dolzina > naj_dolzina:
            naj_dolzina = dolzina
    else:
        dolzina = 0

print("Najdaljše zaporedje dni z negativno temperaturo je dolgo", naj_dolzina)
```

Najdaljše zaporedje dni z negativno temperaturo je dolgo 4

The important thing here is “else”: it is aligned with the first if. The length of the sequence is reset when the temperature is not negative. If we had written this instead:

```
dolzina = 0
naj_dolzina = 0
for vrstica in open("december.txt"):
    temp_c = int(vrstica)
    if temp_c < 0:
        dolzina = dolzina + 1
        if dolzina > naj_dolzina:
            naj_dolzina = dolzina
    else:
        dolzina = 0

print("Najdaljše zaporedje dni z negativno temperaturo je dolgo", naj_dolzina)
```

Najdaljše zaporedje dni z negativno temperaturo je dolgo 8

We would reset the length of the (current) sequence when it did not exceed the length of the longest one, but not, when a non-negative temperature is encountered. However, since the extended sequence would always exceed the length of the longest one, this program essentially counts how many days with negative temperature we have.

0.9 Skill: remembering the previous one

For the last exercise, let's write a program that prints the maximum temperature drop over two consecutive days. In the case of the data prepared by the fortune-tellers, this is 27, because the temperature will one day (supposedly) will drop from 7 to -20. The loop will have to remember the biggest drop. But in addition every new number we have, has to be subtracted from the one we had before.

```
naj_padec = 0
for vrstica in open("december.txt"):
    danes = int(vrstica)
    padec = vceraj - danes
    if padec > naj_padec:
        naj_padec = padec

print("Največji padec:", naj_padec)
```

```
-----
NameError                                Traceback (most recent call last)
Cell In[38], line 4
      2 for vrstica in open("december.txt"):
      3     danes = int(vrstica)
----> 4     padec = vceraj - danes
```

19

```
5     if padec > naj_padec:
6         naj_padec = padec
```

```
NameError: name 'vceraaj' is not defined
```

Well, yes, sort of, but as I said, we need to know yesterday's temperature as well as today's. How do we get to it? Let's remember it at the end of the loop.

```
naj_padec = 0
for vrstica in open("december.txt"):
    danes = int(vrstica)
    padec = vceraj - danes
    if padec > naj_padec:
        naj_padec = padec
    vceraj = danes

print("Največji padec:", naj_padec)
```

```
-----
NameError                                Traceback (most recent call last)
Cell In[39], line 4
      2 for vrstica in open("december.txt"):
      3     danes = int(vrstica)
----> 4     padec = vceraj - danes
      5     if padec > naj_padec:
      6         naj_padec = padec
```

```
NameError: name 'vceraaj' is not defined
```

the last line of the loop happens just before the next round of the loop, before the next line is read. Which is now today will be the next moment yesterday. The program still does not run because the variable vtomorrow does not yet exist in the first round of the loop. How to solve this problem, we'll find out in a lecture or two (to be precise: in a lecture and in two and in three ... there will be a whole bunch of different ways). Today, let's solve it handily: to begin with, let's say that yesterday's temperature was -274.

```
naj_padec = 0
vceraaj = -274
for vrstica in open("december.txt"):
    danes = int(vrstica)
    padec = vceraaj - danes
    if padec > naj_padec:
        naj_padec = padec
    vceraaj = danes

print("Največji padec:", naj_padec)
```

```
Največji padec: 27
```

In the hope that fortune-tellers know something about physics, they will not predict the temperature for the first day, the lower than absolute zero. The temperature on the first day will therefore certainly be more than -

274, so the "drop" will be negative (because the temperature will rise, not fall), so it will not be less than 0, the if condition will not be met and that's it, what will happen in the first round of the loop is that the temperature of the first day will be stored in today. See we have won.

0.9.1 Appendix: subtracting negative numbers

Students like to be puzzled by this: what if the temperature was -2 yesterday and -5 today? Will the fall be calculated correctly? Of course. $-5 - (-2) = -5 + 2 = -3$, as it should be.