02 datoteke, zanke, pogoji

January 28, 2024

0.1 Prva zanka

Ponovimo, še enkrat napišimo program, ki mu vnesemo temperaturo v Celzijevih stopinjah in nam izpiše temperaturo v Fahrenheitih.

[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
```

Razdelili smo ga v dva dela in mednju postavili prazno vrstico. Prvi je kratek - le vrstica, ki priskrbi podatke za drugi del, ki računa in izpisuje.

Zdaj pa si zastavimo višje cilje: v datoteki "temperature.txt" imamo podatke o napovedani temperaturi v Radovljici za teden od 2. do 6. oktobra 2023. Vsebina datoteke takšna:

```
24
18
15
16
18
```

Naša naloga je izvesti program za vsako od teh temperatur. Kot da bi jih vpisovali ročno, le da jih bo bral iz datoteke. Želeli bi torej takšen izpis

```
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
```

Program moramo spremeniti tako, da bo ponavljal drugi del *za vsako* temperaturo iz datoteke. To naredimo tako:

[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
```

Zadnje tri vrstice so take kot prej, novost je prva.

`open` je funkcija. (Kako vemo? Po tem, da jo kličemo. Tako kot smo pisali oklepaje za `input`, `int` in `print`, jih tu pišemo za `open`, torej je `open` funkcija. Če nekemu imenu sledijo oklepaji, to *vedno* pomeni klic funkcije.) Kot argument ji, očitno podamo ime datoteke. Kaj počne, kaj vrne?

Funkcija `open` "odpre" datoteko. Če bi šlo za Word, Excel ali, uh, na primer VLC, bi to pomenilo, da se pokaže besedilo, odpre preglednica, začne predvajati film. V programskih jezikih pa funkcije `open` in podobne (v različnih programskih jezikih imajo lahko različna imena) zgolj poiščejo (ali ustvarijo) datoteko in se pripravijo na njeno branje (ali pisanje). Vem, še vedno je malo abstraktno, a tako bo ostalo. Funkcija `open` vrne nekaj, neko reč, stvar, objekt, ki predstavlja datoteko. Vrne "škatlico", ki ni, kot smo doslej vajeni, `int` ali `float` ali `str` temveč nekaj četrtega, datoteka. (Kdor pričakuje, da bomo povedali ime podatkovnega tipa, na primer `file`, čaka zaman. Ime podatkovnega tipa je čudno, pa še odvisno je od vrste datoteke. Poleg tega pa tega imena nikoli ne potrebujemo. Tako nikoli, da vaš predavatelj za Programiranje 1 niti ne ve, kako se ta tip - oziroma različice teh tipov - v resnici imenujejo.)

Ko smo prejšnji teden spoznali `int`, `float` in `str`, smo rekli, da se z rečmi različnih tipov dajo početi različne stvari. Nize lahko seštevamo in jih množimo s celimi števili. Števila lahko seštevamo, množimo, delimo in še kaj. Kaj pa lako počnemo z datotekami? Bore malo. Seštevati in odštevati jih ne moremo. Za zdaj bomo spoznali le eno operacijo, ki jo zmorejo datoteke: če jih znamo vprašati, vračajo zaporedne vrstice datoteke. Najprej prvo, naslednjič drugo, potem tretjo in tako naprej.

Kako vprašati? Tako kot smo napisali zgoraj: `for vnos in open("temperature.txt"):`. Videvši to, bo Python od datoteke zahteval prvo vrstico. Njeno vsebino bo zapisal v spremenljivko z imenom `vnos` in izvedel vrstice, ki so zapisane pod `for` in zamaknjene. Ko se izvedejo, zahteva od datoteke naslednjo vrstico, njeno vrednost spet priredi spremenljivko `vnos` in spet izvede vrstice. In tako naprej, do konca datoteke.

Temu, kar počnemo tu, računalnikarji rečemo *zanka* (*loop*). Ime mogoče ni najbolj posrečeno, ampak kar je, je. Beseda *zanka* se vedno nanaša na neko ponavljanje. Zanke imajo *glavo*, v našem primeru vrstico `for vnos in open("temperature.txt"):`, ki vodi ponavljanje - more spreminja vrednost kakšne spremenljivke (v tem primeru `vnos`) in, predvsem, določa, kdaj se bo reč končala (v tem primeru se konča, ko je datoteka prebrana do konca). Glavi zanke sledi, kot lahko ugane vsak, ki je že kdaj videl kako žival, *telo*. To so vrstice, ki jih zanka ponavlja.

Python ima dve vrsti zank (nekateri imajo še tretjo, vendar je manj uporabna). Tej tu rečemo zanka *for*, ona, druga, ki nas čaka kasneje, bo *while*. Glava zanke `for`, torej, ima obliko `for <ime-spremenljivke> in <neka-stvar-ki-zna-vračati-elemente>:`. Na koncu glave je vedno dvopičje. Telo, ki sledi v naslednjih vrsticah in mora biti zamaknjeno. Če je potrebno po zanki urediti še kaj, nadaljujemo brez zamika.

Kar poglejmo.

[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!")
```

```
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
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
16 C je 60.8 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
13 C je 55.4 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
```

```
15 C je 59.0 F
Da, dragi moji, tako toplo bo prihodnji teden v Radoljci
10 C je 50.0 F
Da, dragi moji, tako toplo bo prihodnji teden v Radoljci
12 C je 53.6 F
Da, dragi moji, tako toplo bo prihodnji teden v Radoljci
20 C je 68.0 F
Da, dragi moji, tako toplo bo prihodnji teden v Radoljci
10 C je 50.0 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
```

[5]:
```
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")
```

```
15 C je 59.0 F
Da, dragi moji, tako toplo bo prihodnji teden v Radoljci
```

V prvem primeru je zadnji `print` izven zanke, zato se ne ponavlja, temveč izvede po njej. V drugem primeru je znotraj zanke in se bo ponovil za vsako vrstico datoteke.

Najzanimivejši je, kajpada, zadnji, v katerem smo postavili tudi prvi `print` ven iz zanke. Zato se izvede le enkrat. Katero temperaturo izpiše? Zadnjo. Znotraj zanke spreminjamo vrednost `temp_c` in `temp_f`. Po zanki sta takšni, kot smo ju nastavili nazadnje.

Zamiki so lahko poljubni. Upam pa, da se strinjamo, da tole

[6]:
```
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
```

ni prav posrečeno, čeprav deluje. (Sploh pa bo neposrečeno, ko bomo nekoč pisali zanke znotraj zank, pa še kaj drugega bo treba zamikati in s tako majhnim zamikom bodo reči hitro postale nepregledne. Tudi tole ni prav posrečeno:

[7]:
```
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
```

Zategadelj dogovor: umikamo za štiri presledke. Če ima kdo prijatelja, ki zagovarja zamikanje s tabulatorji … Ne. Python bo to sicer toleriral, vendar je za zamike v Pythonu priporočena uporaba presledkov. Štirih.

V vsakem primeru pa morajo biti zamiki konsistentni znotraj bloka.

[9]:
```
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!")
```

```
  File <tokenize>:4
    print(temp_c, "C je", temp_f, "F")
    ^
IndentationError: unindent does not match any outer indentation level
```

To ne deluje, ker četrta vrstica ni ne v zanki ne izven nje. Python vidi, da je zamik manjši in

pričakuje, da bo enak nekemu zunanjemu zamiku (“outer indentation level”), ki je v tem primeru 0.

[10]:
```
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!")
```

```
  Cell In[10], line 4
    print(temp_c, "C je", temp_f, "F")
    ^
IndentationError: unexpected indent
```

To pa ni v redu, ker Python ne ve, čemu smo četrto vrstico nenadoma, brez razloga, umaknili bolj kot tretjo.

0.2 Zanka z računom

Pretvarjanje iz Celzijev v Fahrenheite je klasična začetna naloga iz programiranja in, priznajmo, nekoliko nesmiselna. Koga brigajo Fahrenheiti. (Razen Američanov.) Vsaj mene zanima predvsem, ali bo prihodnji teden v Radovljici toplo (in bom obžaloval, da moram predavati, namesto da bi si privoščil še kak jesenski dan dopusta na kolesu) ali bo mraz. Se pravi, zanima nas povprečna temperatura v prihodnjem tednu. Izračunajmo jo!

Kar. No, dajte, na pamet. Da se spomnimo, temperature so

```
24
18
15
16
18
```

Resno mislim. Izračunajte.

OK, če ste se lotili, kaj ste počeli? Tole: 24 in 18 je 42. 42 in 15 je 57. 57 in 16 je 73. 73 in 18 je 91. 91 / 5 je 18.2. Drži?

Računalnik mora narediti natančno isto. Iti mora čez temperature. Za vsako temperaturo (aha: `for vnos in open("temperature.txt"):`) mora k dosedanji vsoti prišteti to temperaturo. Takole?

[11]:
```
for vnos in open("temperature.txt"):
    vsota = vsota + vnos
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[11], line 2
      1 for vnos in open("temperature.txt"):
----> 2     vsota = vsota + vnos
```

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

No, skoraj takole. Ideja je čisto prava: v vsakem koraku zanke, moramo spremenljivki `vsota` prirediti vrednost, ki jo dobimo tako, da k prejšnji vrednosti `vsota` prištejemo vrednost iz vrstice datoteke. Python se pritožuje le, da ne pozna spremenljivke z imenom `vsota`.

To se zgodi, ker že v prvem krogu poskušamo prištevati k `vsota`, še preden ta spremenljivka obstaja. To je lahko rešiti: pred zanko jo nastavimo na `0`. Potem pa bomo lepo prištevali.

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

Tole pa poznamo. `vsota` je `int`, `vnos` pa besedilo, `str`. `vnos` bo potrebno pretvoriti v številko. (Mimogrede ga še preimenujmo v kaj pametnejšega, recimo, `temperatura` ali `vrstica`.)

[12]:
```
vsota = 0
for vrstica in open("temperature.txt"):
    vsota = vsota + int(vrstica)
```

Pa zdaj? Nič se ni zgodilo?! Je, samo nič se ni izpisalo. Na koncu izpišimo vsoto.

[13]:
```
vsota = 0
for vrstica in open("temperature.txt"):
    vsota = vsota + int(vrstica)
print(vsota)
```

```
201
```

Mi pa, seveda, potrebujemo povprečje. Če vemo, da vsebuje datoteka natančno 5 temperatur, lahko seveda izpišemo `vsota / 5`. Če želimo biti splošnejši, pa bomo sproti, ob seštevanju, še šteli dneve.

[14]:
```
vsota = 0
dni = 0
for vrstica in open("temperature.txt"):
    vsota = vsota + int(vrstica)
    dni += 1
print(vsota / dni)
```

```
15.461538461538462
```

Ali, če hočemo počasneje:

[15]:
```
vsota = 0
dni = 0
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    vsota = vsota + temp
```

```
    dni += 1
print(vsota / dni)
```

```
15.461538461538462
```

To je to.

Ampak, no, to ni to. Pravzaprav me mogoče ne zanima povprečna temperatura, temveč bi rad vedel, ali bo temperatura kdaj padla pod 20 stopinj. Ali pa, mogoče, na kateri dan bo najvišja. Morda pa bi me navdihnilo ocenjevanje v slogu smučarskih skokov in bi želel vedeti, kakšna bo povprečna temperatura, če ne upoštevam najtoplejšega in najhladnejšega dne …

0.3 Pogojni stavki

Začnimo z manj ambicioznim ciljem. Ker je znano, da človeka zebe, ko je temperatura manjša ali enaka 17 stopinj, torej takšnih temperatur nočemo videti, izpišimo vse temperature, večje od 17.

Hm. Za zdaj znamo samo izpisati vse temperature.

[16]:
```
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    print(temp)
```

```
24
18
15
16
18
13
15
15
10
12
20
10
15
```

Naloga, ki smo si jo zastavili, pa je izpisati le tiste izmed teh številk, ki so večje od 17. Tisti `print` torej, se ne sme zgoditi vedno, temveč le, če velja `temp > 17`.

Nič lažjega.

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

```
24
18
18
20
```

Tako kot s `for` od Pythona zahtevamo, da nekaj ponavlja, z `if` zahtevamo, da nekaj stori le, če je izpolnjen določen pogoj.

`for`-u (in kasneje `while`-u) rečemo zanka. Tudi `if`-u nekateri rečejo zanka, vendar le študenti-začetniki. :) `if` (skupaj s tem, kar mu sledi), je *pogojni stavek*.

Tako vrstico s `for`, moramo tudi vrstico z `if` zaključiti z dvopičjem, slediti pa ji mora ena ali več zamaknjenih vrstic. Ker je že `if` sam zamaknjen za štiri presledke (ker je znotraj `for`), bo `print`, ki je znotraj `if`, zamaknjen za osem presledkov.

(

Medklic za družbo- ali celo jezikoslovno naravnane bralce. Ob pisanju gornjega stavka sem sprva pozabil zadnjo vejico, tisto pred "zamaknjen". Pri teh zamikih in odmikih gre za praktično enako reč, kot so v jeziku podredni stavki, ki so lahko tudi podrejeni, tako kot je recimo tale, drugim podrednim stavkom, mar ni tako? Se prvi

```
Pri teh zamikih in odmikih gre za praktično enako reč,
    kot so v jeziku podredni stavki,
        ki so lahko tudi podrejeni,
            tako kot je recimo tale,
        drugim podrednim stavkom,
mar ni tako?
```

V slovenščini začetek in konec podrednega stavka nakažemo z vejico (ki žal ne pove, ali gremo noter ali ven), v Pythonu pa to naredimo z zamiki in, zavoljo preglednosti, še dvopičjem ob zamiku.

)

Pa če bi radi zgolj prešteli, koliko bo toplih dni? V primeru teh podatkov so trije; želimo torej zgolj izpisati `3`, ne pa posamičnih temperatur.

V tem primeru `print` nadomestimo s preštevanjem.

[18]:
```
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
```

Tole se ne sliši prav lepo. Slovenščino boste popravili sami, za nalogo. Tule si zastavimo drug cilj: povedati želimo, da bo toplih dni 3 od 5.

Za to, moramo poleg toplih dni šteti še *vse dni*. To pa že znamo.

[19]:
```
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
```

Vidimo odmike? Vrstica `toplih = toplih + 1` je znotraj `if`, zato se izvede le za tiste vrstice datoteke, ki vsebujejo temperature večje od 17 Vrstica `dni = dni + 1` pa ni znotraj `if`, je pa znotraj `for`. Zato se izvede za vsako vrstico datoteke. Če bi jo izmaknili še bolj, ven iz zanke, pa bi se izvedla le enkrat.

[20]:
```
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 Sicer

Naslednja naloga: nekoga zanima, bo li kdaj v prihodnjem tednu temperatura vsaj 20 stopinj. Želeli bi torej program, ki izpiše bodisi "Da, v prihodnjem tednu bo tudi kak topel dan." ali "Ne, prihodnji teden bo ena žalost."

Kmalu se bomo naučili bolj pravega način reševanja te naloge, vendar smo ji kos tudi s tem, kar znamo doslej. Preprosto preštejmo, koliko dni bo temperatura vsaj 20 stopinj in potem preverimo, ali je takih dni več kot 0.

[21]:
```
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.
```

Program deluje, vendar bo deloval le še kak teden, morda dva. Pač dokler bo ARSO poročal, da bodo "temperature relativno visoke za ta letni čas". Čim bodo padle pod dvajset, program ne bo več izpisal ničesar.

No, da seveda.

[22]:
```
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.
```

Ni kul. Dvakrat.

S pogojnimi stavki včasih zapovemo, naj se nekaj zgodi le v določenem primeru. Pogosto pa z njimi določamo dva scenarija: če pogoj drži, stori eno, sicer stori nekaj drugega. Ta primer je že tak: če je kak dan `nad_20`, naj izpiše, da topli dnevi bodo, sicer naj izpiše, da jih ne bo. V vseh normalnih programskih jezikih sme `if`-u slediti `else`.

[23]:
```
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.
```

Tudi `else` ima, tako kot `if`, na koncu vrstice dvopičje, sledi ena ali več zamaknjenih vrstic.

[24]:
```
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.
```

Za razliko od `if` pa `else` seveda nima pogoja. Pogoj je napisan že zgoraj, v `if`. Vrstice, ki sledijo `else`, se bodo pač izvedle takrat, ko je pogoj `neresničen`.

Če imamo več kot dva lepa dneva, bomo rekli, da bo teden zelo lep in škoda, da ne moreš vzeti dopusta. Sicer, če bo lep dan en sam, posvarimo, da bo en sam in da se ga splača izkoristiti. Sicer stokamo.

[25]:
```
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!
```

Deluje, ampak vidimo problem? Začasno spustimo temperaturo, namesto `> 19` preverjajmo `> 10`.

[26]:
```
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!
```

Drugi `if` preverja, ali imamo kak topel dan. Vendar bi moral to početi le, če ni ravno prej izpisal, da bosta (vsaj) dva. Pravilno (ekhm, pravilneje) je tako:

[27]:
```
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!
```

Razumemo. Predstavljamo pa si tudi problem: tile zamiki se znajo kar pošteno zagnezditi - if znotraj if znotraj else znotraj for … Temu se kaže izogniti. Pa še nekaj: ta program je oblikovan tako, da imamo dve možnosti (topla dneva sta vsaj dva ali pa manj kot dva), pri čemer se druga možnost potem deli na podmnožnosti (je vsaj kak ali ni nobenega). Taka struktura da pravilne rezultate, ni pa skladna s tem, kako v resnici gledamo na možnosti. Vsaj jaz tu vidim tri "enakovredne" alternative: "vsaj dva", "vsaj eden", "nobeden". Deljenje ene možnosti na dve podmožnosti … če tega nimam v glavi, naj tudi ne bo v programu.

Ker so take situacije pogoste, ima Python tudi možnost `elif` - ki je okrajšava za *else if* (in je tudi razlog, da sem v besedilu, ki je opisovalo, kaj hočemo storiti, uporabil slogovno nerodno zvezo "sicer če").

[28]:
```
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!
```

Da preverimo, da res deluje, dvignimo želeno temperaturo nazaj na 20 stopinj.

[29]:
```
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 Drugi operatorji za primerjanje

Nekje proti začetku prejšnjega razdelka sem se pritoževal, da mi nekaj ni kul, konkretno, dva `if`-a z nasprotnima pogojema in takrat smo uvedli `else`. Rekel pa sem, da mi "dvakrat ni kul". Drugo, kar mi ni kul v vseh programih iz gornjega razdelka, je, da se sprašujemo, ali je temperatura vsaj `20` stopinj in to zapišemo kot `if temp > 19`. Podobno smo se spraševali, ali imamo vsaj tri tople dni in napisali `if nad_20 < 2`. Da ne govorimo o norem pogoju, ki je preverjal, ali je `nad_20` enak nič tako, da se je vprašal `if nad_20 < 1`.

Ko programiramo, je dobro, da pišemo, kar mislimo. (Po drugi strani pa je del učenja programiranja seveda tudi, kako je potrebno misliti, da se bo to potem dalo sprogramirati.) Če rečemo, da mora biti temperatura vsaj 20, to ni "več kot 19" temveč "več ali enako 20". Ali, kot bi rekli matematiki, ne `temp > 19`, temveč `temp  20`. Tudi računalnikarji bi lahko rekli tako; Python bi brez težav sprejemal simbol , vendar ga je nepraktično natipkati, ker ga tipkovnice pač nimajo. Zato namesto pišemo `>=`.

Za primerjanje imamo torej operatorje `<`, `<=`, `>` in `>=`. Z njimi lahko program, ki ga vrtimo zadnje pol ure, polepšamo v

[30]:
```
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!
```

To je že boljše, vendar bi v drugem pogoju pravzaprav radi vprašali, ali imamo en tak dan. Mogoče tako?

[31]:
```
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 '='?
```

Očitno ne. Pogoj `nad_20 = 1` mu ni všeč. V tem vidi prirejanje, ne primerjanja, potem pa se seveda pritoži, ker prirejanje nima kaj iskati znotraj pogoja. Nekateri (ampak redki!) jeziki "uganejo", da v tem primeru z "=" mislimo primerjanje, večina pa ima za prirejanje in primerjanje različne simbole. Python (in tudi večina drugih) za primerjanje uporablja dvojni enačaj.

[32]:
```
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 Primerjanje nizov

V tem razdelku nimamo napisati kaj veliko, vendar vseeno naredimo vmesni naslov, da ga poudarimo: z operatorji `<`, `<=`, `>`, `>=`, `==` in `!=` lahko primerjamo tudi nize. Nize primerja *dokaj* tako, kot bi človek pričakoval: po abecedi. Ana je "manj" od "Angele", a več od "Amelije".

Samo *dokaj* pa zato, ker so velike črke po abecedi pred malimi. Ker so številke pred črkami. Ker je narekovaj pred številko. Zaviti oklepaji so za črkami, oglati pa so po "abecedi" med velikimi in malimi črkami. Šumniki so … kar nekje, vsekakor pa za črkami.

Te stvari imajo razloge. In imajo tudi rešitve. Vendar o tem kasneje. Za zdaj bodi dovolj, da vemo, da smemo nize primerjati in bodo rezultati (relativno) smiselni.

0.7 Klasika: maksimum in minimum

Napišimo program, ki izpiše najvišjo temperaturo.

Predstavljajmo si, da bi nam nekdo bral te številke. Kaj bi si ponavljali v glavi? Najvišjo, ki smo jo slišali doslej. Vsako naslednjo številko, ki jo slišimo, primerjamo z najvišjo doslej in če je višja od najvišje, si jo zapomnimo (in pozabimo prejšnjo). Tako nekako:

[33]:
```
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
```

Zgodilo se nam je, kar se nam je že: spremenljivke `n jvisja_doslej` ne moremo uporabiti (se primerjati z njo) znotraj zanke, če je ne nastavimo pred zanko. Mimogrede jo bomo še preimenovali v `najvisja`. Daljše ime, `najvisja_doslej` je služilo bolj … didaktičnom namenom, da začetnik lažje prebere. Zdaj pa raje skrajšajmo, da bo manj tipkanja. :)

Hm, kakšno vrednost pa ji bomo dali pred zanko? Za zdaj, z našim znanjem, naj bo to kar nekaj absurdno nizkega, tako da bo že naslednja temperatura višja. Vsaj kar se tiče temperatur, smo lahko varni: fiziki nas uče, da temperatura ne more biti nižja od 0 K, torej bo -274 varna začetna vrednost.

[34]:
```
najvisja = -274
for vrstica in open("temperature.txt"):
    temp = int(vrstica)
    if temp > najvisja:
        najvisja = temp

print(najvisja)
```

```
24
```

Najnižjo napovedano temperaturo izračunamo enako, le obratno. Začetna vrednost pa je lahko milijon. Enkrat drugič se bomo naučili to narediti pravilneje.

Da torej ne bi ponavljali v bistvu istega programa, se poskusimo z nečim težjim: poleg najnižje temperature nas zanima tudi zaporedna številka dne s to temperaturo.

Spet pomislimo, kako bi to naredili, če bi nam nekdo govoril bral številke. Poleg najnižje številke bi si morali zapomniti še dve števili: zaporedno številko tekočega dneva (to že znamo: spomnite se, kako smo šteli dni, ko smo izpisali, koliko dni *od kolikih* bo toplo), poleg tega pa si moramo istočasno z najnižjo številko zapomniti še takratno številko tekočega dneva.

[35]:
```
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 Opazovanje zaporedij

Tri stare vedeževalke so staknile glavo in napovedale decembrske temperature. Ob pomoči vnuka so jih vtipkale v datoteko z imenom "december.txt":

```
5
4
-1
-6
-8
2
5
-6
-8
-12
-15
6
7
-20
2
3
```

Za novo vajo iz razmišljanja o zankah in spremenljivkah napišimo program, ki pove, kakšno bo (predvidoma) najdaljše zaporedje dni, ko bo temperatura pod ničlo.

Če bi se to šli, tako kot smo to počeli na predavanjih - jaz govorim številke, študenti sledijo in na koncu odgovorijo - bi se morali zapomnjevati dve stvari: kako dolgo je trenutno zaporedje dni z negativno temperaturo in kako dolgo je najdaljše zaporedje. Vsakič, ko slišimo številko preverimo, ali je negativna ali ne. Če ni, potem "resetiramo" dolžino zaporedja na 0. Če je negativna, pa k dolžini zaporedja prištejemo 1 in preverimo, ali smo slučajno dobili zaporedje, ki je daljše od najdaljšega.

Če si (upam) predstavljamo, kaj bi morali delati v glavi, potem znamo taisto (upam) ubesediti v Pythonu.

[36]:
```
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
```

Pomembna stvar tule je `else`: ta je poravnan s prvim `if`. Dolžino zaporedja resetiramo, ko temperatura ni negativna. Če bi namesto tega pisali:

[37]:
```
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
```

bi dolžino (trenutnega) zaporedja resetirali, kadar to ne bi preseglo dolžine najdaljšega, ne pa, kadar bi naleteli na nenegativno temperaturo. Ker pa bi podaljšano zaporedje *vedno* preseglo dolžino najdaljšega (razmislite zakaj!), ta program v bistvu prešteje, koliko je dni z negativno temperaturo.

0.9 Spretnost: pomnjenje prejšnjega

Za zadnjo vajo napišimo še program, ki izpiše največji padec temperature v dveh zaporednih dneh. V primeru podatkov, ki so jih pripravile vedeževalke, je to 27, saj bo temperatura nekoč (menda) padla s 7 na -20.

Spet razmislimo, kako bi to teklo v živo: jaz govorim številke, študenti morajo na koncu povedati, kakšen je bil največji padec. Očitno bi si morali zapomniti največji padec doslej. Poleg tega pa bi vsako novo številko, ki jo slišite, odšteli od tiste, ki smo jo slišali pred njo.

[38]:
```
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
```

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

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

No, ja, tako nekako, ampak, kot rečeno, poleg današnje temperature moramo poznati včerajšnjo. Kako do nje? Zapomnimo si jo na koncu zanke.

[39]:
```
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 'vceraj' is not defined
```

Zadnja vrstica zanke se zgodi tik pred naslednji krogom zanke, pred branjem naslednje vrstice. Kar je zdajle `danes` bo v naslednjem trenutku `vceraj`.

Program še vedno ne deluje, ker v prvem krogu zanke še ne obstaja spremenljivka `vceraj`. Kako lepo rešiti ta problem, bomo izvedeli čez predavanje ali dve (točneje: čez predavanje in čez dve in čez tri … različnih načinov bo cel kup). Danes ga rešimo rokohitrsko: za začetek recimo, da je včerajšnja temperatura -274.

[40]:
```
naj_padec = 0
vceraj = -274
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)
```

```
Največji padec: 27
```

V upanju, da vedeževalke vedo kaj o fiziki, za prvi dan ne bodo napovedovale temperature, manjše od absolutne ničle. Temperatura prvega dne bo torej gotovo več kot -274, torej bo "padec" negativen (ker bo temperatura narasla, ne padla), torej ne bo manjši od 0, pogoj v `if` ne bo izpolnjen in vse, kar se bo zgodilo v prvem krogu zanke, je, da se bo temperatura prvega dne shranila v `danes`. Pa smo zmagali.

0.9.1 Dodatek: odštevanje negativnih števil

Študente rado zbega tole: kaj, če je bila temperatura `vceraj` -2, `danes` pa -5? Se bo padec izračunal pravilno? Seveda. `-5 - (-2) = -5 + 2 = -3`, kot mora biti.