Napake iz P 1

Nekaj tipičnih napak/pomanjkljivosti od lani

Premalo "komunikacije"

- Premalo sprašujete
- Sodelujete v forumih
- Ni vas na govorilne ure
- Premalo ste pripravljeni na vaje
  - Čas vaj se izkoristi, da se sprašuje o tistem, kar se je med pripravo (doma) zataknilo
  - Morda tudi o prijemih, ki jih niste slišali na predavanjih, pa ste jih videli v "tuji" kodi
  - …
  - Čas vaj ni čas osnovnega spoznavanja snovi

Izpisovanje v metodah

- Za javljanje napak

```
print("Število ne sme biti 0!")

# nasploh "idealno", če potem kar nadaljujemo …
-------------

def glavna(dat_temp,dat_odst):

        #preverimo ali datoteke obstajajo

    if not os.path.isfile(dat_temp+'.txt'):

        print('Datoteka z temperaturami NE obstaja!')

        return
```

- Uporabljajo se izjeme!
  - S tem ima tisti, ki metodo uporablja, možnost "popravnega izpita"

Izpisovanje v metodah

- Za "vračanje" rezultatov
  - Izračunaj vsoto števil v seznamu
  - Z izpisovanjem
  - Z return

```
def skVsota(sez) :
    …
    print(vsota)


print(skVsota([2, 3, [4, 5]]))
```

Anglovenščina oz. Slogleščina

- Zelo radi "mešate" jezike

- `def main(dat_temp,dat_odst):`
- `file_temp=open(dat …`

```
def clen(digit = 1000):
    '''vrne prvi clen Fibonaccijevega zaporedja, ki
    vsebuje vneseno stevilo stevk'''
    digit = int(digit)
    if digit == 1:
        return 1
    a = 0
    b = 1
    term = 0
```

Neustrezno komentiranje

- Premalo komentarjev
  - Komentirati je potrebno ključne dele kode
  - Povemo idejo!
  - Pisanje komentarjev vnaprej/sproti
- Preveč komentarjev
  - Ne vsake "obskurne" podrobnosti
  - "Ciljna publika"
- Prevodi kode
  - V komentar ne napišemo "prevod" kode v slovenščino, ampak KAJ/ZAKAJ smo nekaj naredili!

```
else: #sicer (če pogoj ni bil izpolnjen)
```

```
deliZ=n - 1 # začetno številko pomanjšamo za 1
while deliZ > 0: # zanka, ki bo delovala, dokler
                 # bo število s katerim bomo delili večje od 0
if n % deliZ == 0: # pogojni stavek, ki preveri, če
                   # je število delitelj začetnega števila
vsotaDeliteljev+=deliZ # prištejemo delitelja
deliZ -= 1 #delitelja pomanjšamo za 1
```

```
f=open(dat)    # odpremo datoteko
i=1          # spremenljivka i šteje prebrane vrstice
obiskana=0   # spremenljivka, ki šteje,
             # koliko polj je skakač že obiskal
v=f.readline()  #preberemo vrstico
if '\n' in v:
   v=v[:-1]  # če vrstica vsebuje znak '\n',
             # je na zadnjem mestu in
             # ga odstranimo, saj ga ne potrebujemo
```

Imena spremenljivk

- Uporabljamo imena, ki nekaj pomenijo
- Zakaj `i`, zakaj ne `stevecPrebranihVrstic`

Zanka for

- `for i in range(len(sez)) :`
  - `x = sez[i]`
  - `… Stavki, kjer i nikoli ne rabimo, le x`
- Zakaj ne:
  - `for x in sez :`

Domače/seminarske naloge - poročila

- Kaj je ideja
- Smiselni testni primeri
- Oblikovanje

Znamo izluščiti idejo?

- Metoda za vhodni podatek dobi ime datoteke, na kateri so zapisani premiki skakača. Predpostavimo, da datoteka v vsakem primeru obstaja in da pri odpiranju datoteke ne pride do težave. V prvi vrstici datoteke morajo biti zapisana tri števila - velikost šahovnice (n) in koordinati začetnega polja (dve števili od 1 do n), v naslednjih vrsticah pa so opisi premikov – število polj, za katero se skakač premakne v vodoravni smeri in število polj, za katero se premakne v navpični smeri. Števila so med seboj ločena s presledkom. Če so števila negativna, pomeni, da je to premik navzdol ali levo, če pa pozitivna pa navzgor ali desno. Metoda vrže izjemo in pove v kateri vrstici je prišlo do katere napake v primeru, če na datoteki niso cela števila, če v kakšni vrstici manjka podatek oziroma jih je preveč, če poteza ni v skladu z dovoljenimi premiki skakača ali pa, če podatki predstavljajo nepravilne koordinate. V primeru, da zaporedje potez predstavlja skakačev obhod, metoda vrne »True«, drugače pa »False«. Če zaporedje potez predstavlja pravilne premike, vendar je število potez manjše od n×n, to pomeni, da skakač ne more obiskati vseh polj in metoda vrne »False«.

Kaj pa tu?

- Metoda za vhodni podatek dobi ime datoteke, na kateri so zapisani premiki skakača.
- Ustvarimo tabelo, ki ponazarja, ali so polja na šahovnici že obiskana
- Zapomnimo si trenutni položaj skakača
- Za vsako vrstico na datoteki
  - Glede na prebrane podatke izračunamo nov položaj skakača
  - Če je pri tem prišlo do težav (napačni podatki …) vržemo ustrezno izjemo
  - Če je tisto mesto že bilo obiskano, zaključimo z odgovorom false
  - Drugače pa si v tabeli zapomnimo, da smo mesto obiskali
- Če smo po koncu obdelave datotek našteli n x n obiskanih polj, odgovorimo True (gre), sicer pa False (ni šlo)

Ali še bolje

- Osnovna ideja je:
  - Vodili bomo trenutni položaj skakača
  - Ob vsaki potezi bomo preverili, če gre za legalno potezo (znotraj polja, pravilni podatki …) in če nas ne postavi na že obiskano polje
  - Če poteza ni legalna, takoj zaključimo
  - Ko smo preverili vse poteze (in so bile vse legalne) smo morali obiskati n2 polj
- Podrobneje:
  - Prejšnja prosojnica

Testni primeri

- "ugodni" izidi
- "neugodni" izidi
  - Zaradi ponovljenega skoka
  - Zaradi neobiskanega polja
- Napačni podatki (za preverjanje proženja izjem …)
- "Robni primeri"
  - Šahovnica manjša kot 3 x 3
- Testne primere je dobro (praktično nujno) pripraviti in premisliti vnaprej