07 Oblikovanje nizov

January 28, 2024

1 Oblikovanje nizov

Pri izpisovanju smo bili doslej omejeni na dokaj zoprni `print`. Za začetek nam je zadoščal, kmalu pa postane neroden, saj nekaterih stvari z njim preprosto ni mogoče (preprosto) narediti. Za začetek se je potrebno z njim pogajati, da ne gre po vsakem izpisu v novo vrsto in da ne dodaja presledkov med reči, ki jih izpisujemo.

Če smo imeli slovar, v katerem je pisalo, koliko kilometrov je nekdo prevozil s katerim izmed svojih koles,

[1]:
```
razdalje = {'Canyon': 2932, 'Cube': 2939, 'Nakamura': 488, 'Stevens': 0}
```

in hoteli to izpisati, nam je `print` dodajal presledke pred dvopičja.

[2]:
```
for kolo, razdalja in razdalje.items():
    print(kolo, ":", razdalja)
```

```
Canyon : 2932
Cube : 2939
Nakamura : 488
Stevens : 0
```

Če bi želeli poleg tega v oklepjih izpisati, kakšen delež je prevozil s katerim kolesom, bi dobili kaj takole neprebavljivega:

[3]:
```
skupna = sum(razdalje.values())

for kolo, razdalja in razdalje.items():
    print(kolo, ":", razdalja, "(", razdalja / skupna * 100, "%)")
```

```
Canyon : 2932 ( 46.10787859726372 %)
Cube : 2939 ( 46.217958798553234 %)
Nakamura : 488 ( 7.6741626041830475 %)
Stevens : 0 ( 0.0 %)
```

Preveč decimalk, tile grozni presledki … morda pa bi se prileglo tudi kaj poravnavanja, ne?

Slaba novica: `print`-u sicer lahko rečemo, naj ne piše odvečnih presledkov, s poravnavanjem in decimalkami pa se mu res ne da ukvarjati.

Večina besedil, ki jih izpišejo ali kako drugače pokažejo programi, vsebuje fiksno besedilo, v katerega so vstavljeni kaki nizi ali številke. Če računalnik izpiše "Datoteke kraljice.py ni mogoče najti", je vstavljeno besedilo "kraljice.py", ki je najbrž vsakič drugačno, odvisno pač od tega, katero datoteko (neuspešno) iščemo. V "Danes je 60 Fahrenheitov, to je, 16 Celzijevih stopinj" sta vstavljeni števili 60 in 16, ki sta odvisni od tega, v katerem letnem času poženemo program (in koliko smo že uspeli pogreti planet), ostalo besedilo pa je vedno enako.

Večina računalniških jezikov omogoča, da sestavimo nize tako, da napišemo "konstantni" niz, označimo mesta, na katera je potrebno nekaj vstaviti. Potem pa na nek način povemo, kaj naj vstavi. Popularni prevajani jeziki so večinoma povzeli način, ki so si ga izmislili v jeziku C in je malo bolj zoprn, novejši jeziki, predvsem skriptni, ki so bližje spletu, pa navadno omogočajo preprostejše in zmogljivejše načine sestavljanja nizov iz vzorcev.

Eno od načel Pythona je, da se da vsako stvar narediti na en in samo en očiten način. To je dobro zato, ker, posledično, vsi programerji pišemo podobne programe in zato lažje sodelujemo. Prav pri oblikovanju nizov pa se je to sfižilo: nize lahko oblikujemo na tri različne načine. Do tega je prišlo, ker so določene ideje (ne le v Pythonu) zorele z leti, vendar (za)star(el)ih načinov ne moreš kar tako ukiniti, saj bi starejši programi potem ne delali v novejših različicah Pythona. Zaresen programer v Pythonu mora poznati vse tri načine, nam bo zadoščal zadnji, najmodernejši. V resnici tudi jaz uporabljam le še tega.

1.1 f-nizi

Recimo, da imamo

[4]:
```
fahr = 42
celz = (fahr - 32) * 5 / 9
```

Zdaj bi radi sestavili, v bistvu, takšen niz:

[5]:
```
"{fahr} Fahrenheitov je {celz} Celzijev"
```

[5]:
```
'{fahr} Fahrenheitov je {celz} Celzijev'
```

le da želimo, da Python na mesti, ki smo ju označili s `{fahr}` in `{celz}`, vstavi vrednosti spremenljivk `fahr` in `celz`.

To počnejo f-nizi. F-niz je kot običajen niz, le da mu pred prvi narekovaj dodamo črko f (kot *format*). V takšnih nizih imajo zaviti oklepaji poseben pomen: kar je v njih, se (izračuna in) vstavi.

[6]:
```
f"{fahr} Fahrenheitov je {celz} Celzijev"
```

[6]:
```
'42 Fahrenheitov je 5.555555555555555 Celzijev'
```

V neki domači nalogi smo - oh, kako je bilo to nerodno! - pisali

```
from random import randint

a = randint(2, 10)
b = randint(2, 10)
```

```
print(a, "krat", b)
c = input("Odgovor? ")
```

S tem, da najprej izpišemo in potem zahtevmo odgovor, smo se izognili temu, da bi morali mukoma sestavljali niz ob klicu funkcije `input`:

```
a = randint(2, 10)
b = randint(2, 10)
c = input("Koliko je " + str(a) + "x" + str(b) + "?")
```

Zdaj, ko poznamo metodo f-nize, poznamo preprostejši način:

```
a = randint(2, 10)
b = randint(2, 10)
c = input(f"Koliko je {a}x{b}? ")
```

(Mimogrede omenimo starejša načina oblikovanja nizov, ki smo ju omenjali na začetku: v C-jevskem slogu bi pisali `c = input("Koliko je %sx%s? " % (a, b))`, v onem malo novejšem pa `c = input("Koliko je {}x{}? ".format(a, b))`. Obema je skupno to, da so v nizu le označena mesta, kamor je potrebno vstavljati, potem pa moramo na nekem drugem mestu, po koncu niza, našteti stvari, ki jih vstavljamo. Z novejšim slogom lahko z nekimi čudnimi triki v zavite oklepaje dodajamo tudi imena spremenljivk, vendar niti približno tako udobno kot po novem.)

Na ta način ne vstavljamo le števil, temveč karkoli, kar se da izpisati.

[7]:
```
datoteka = "kraljice.py"
f"Datoteke {datoteka} ne najdem"
```

[7]:
```
'Datoteke kraljice.py ne najdem'
```

1.2 Izrazi v f-nizih

V nize ne vstavljamo le vrednosti spremenljivk: v zavite oklepaje lahko pišemo kar cele izraze.

[8]:
```
fahr = 42

f"{fahr} Fahrenheitov je {(fahr - 32) * 5 / 9} Celzijev"
```

[8]:
```
'42 Fahrenheitov je 5.555555555555555 Celzijev'
```

Zdaj pa spet na kolesa!

[9]:
```
skupna = sum(razdalje.values())

for kolo, razdalja in razdalje.items():
    print(f"{kolo}: {razdalja} ({razdalja / skupna * 100} %)")
```

```
Canyon: 2932 (46.10787859726372 %)
Cube: 2939 (46.217958798553234 %)
Nakamura: 488 (7.6741626041830475 %)
Stevens: 0 (0.0 %)
```

Delen uspeh. Znebili smo se presledkov, predvsem pa je postal izpis preglednejši. Kar primerjajmo:

```
prej: print(kolo, ":", razdalja, "(", razdalja / skupna * 100, "%)")
zdaj: print(f"{kolo}: {razdalja} ({razdalja / skupna * 100} %)")
```

Če hočemo še pregledneje, pa lahko napišemo

[10]:
```
skupna = sum(razdalje.values())

for kolo, razdalja in razdalje.items():
    delez = razdalja / skupna * 100
    print(f"{kolo}: {razdalja} ({delez} %)")
```

```
Canyon: 2932 (46.10787859726372 %)
Cube: 2939 (46.217958798553234 %)
Nakamura: 488 (7.6741626041830475 %)
Stevens: 0 (0.0 %)
```

Zdaj je f-niz očitno vzorec, v katerega vstavljamo vrednosti.

Ostane nam še boj z decimalkami.

1.3 Kako oblikujemo reči

Če bi radi povedali, kako naj se izpiše neko število (ali niz ali karkoli že), dodamo izrazu dvopičje in za njim opis formata.

To bomo najpogosteje potrebovali za oblikovanje necelih števil. To gre tako:

[11]:
```
f"{fahr:4.1f} Fahrenheitov je {celz:4.1f} Celzijev"
```

[11]:
```
'42.0 Fahrenheitov je  5.6 Celzijev'
```

Opis formata je tule `4.1f`. Pri tem `4.1` pomeni, da bi radi izpis na štiri mesta, pri čemer naj bo eno rezervirano za decimalko. Za črko `f` si predstavljajte, da pomeni `float`.

Če pustimo za število premalo prostora, ga bo izpis pač zasedel več.

[12]:
```
x = 1234.5678
f"Primer predolgega števila: {x:3.1f}"
```

[12]:
```
'Primer predolgega števila: 1234.6'
```

Zahtevali smo eno decimalko in to smo tudi dobili, vse skupaj pa je širše kot štiri mesta, saj je število pač predolgo.

Določanje števila decimalk je smiselno le pri necelih številih. Če vstavljamo nize ali cela števila, lahko določamo le širino:

[13]:
```
podatki = [
    (74, "Anze", False),
    (82, "Benjamin", False),
    (48, "Cilka", True),
```

```
    (66, "Dani", False),
    (61, "Eva", True),
    (101, "Franc", False),
    ]

for teza, ime, spol in podatki:
    print(f"{ime:10}{teza:6}")
```

```
Anze          74
Benjamin      82
Cilka         48
Dani          66
Eva           61
Franc        101
```

Tu za številom mest nismo dodajali črke `f`, saj ne gre za (necela) števila. Za nizi pa sploh ne. (Obstajajo sicer druge črke, ki bi jih lahko dodali, da Python povemo, za kaj gre, vendar jih smemo tule brez škode pozabiti.)

Vsa imena so izpisana z desetimi znaki; manjkajoči prostor je zapolnjen s presledki. Številke so izpisane s šestimi mesti, manjkajoči prostor spet zapolnjujejo presledki.

Nizi so poravnani na levo - presledki so dodani za njimi. Večina reči, ki jih izpisujemo (nizi in števila namreč niso edino, kar se da izpisati), so poravnani tako. Številke pa so poravnane desno. To lahko spremenimo, če za dvopičje dodamo `<`, `>` ali `^`, s katerimi naročimo, naj bo reč poravnana levo, desno ali na sredino. Ravno obratni izpis kot zgoraj bi dosegli z

[14]:
```
for teza, ime, spol in podatki:
    print(f"{ime:>10}{teza:<6}")
```

```
      Anze74
  Benjamin82
     Cilka48
      Dani66
       Eva61
     Franc101
```

To ni ravno posrečeno. Dodajmo še presledek.

[15]:
```
for teza, ime, spol in podatki:
    print(f"{ime:>10} {teza:<6}")
```

```
      Anze 74
  Benjamin 82
     Cilka 48
      Dani 66
       Eva 61
     Franc 101
```

To je lažje berljivo, še vedno pa je seveda najlepši prvi izpis.

Pred znak, ki določa poravnavo, lahko dodamo simbol, ki naj se uporabi za zapolnjevanje - če seveda nismo zadovoljni s presledkom. Poravnajmo imena levo, števile desno, namesto presledkov pa zapolnimo prazen prostor s pikami.

[16]:
```
for teza, ime, spol in podatki:
    print(f"{ime:.<10}{teza:.>6}")
```

```
Anze…74
Benjamin…82
Cilka…48
Dani…66
Eva…61
Franc…101
```

Namesto pik lahko uporabimo poljuben drug znak. Kadar takole zapolnjujemo prostor s čimerkoli drugim kot s presledki, moramo dodati znake za poravnavanje (`<`, `>` ali `^`), tudi kadar z njimi izberemo takšno poravnavanje, kot bi bilo tudi privzeto (desno za števila, levo za vse drugo) všeč.

V vse detajle ne bomo šli. Določiti je mogoče še veliko reči. Pri številih lahko zahtevamo, naj se vedno izpiše predznak, torej +, kadar je število pozitivno. Za števila lahko določimo, naj se izpišejo dvojiško ali šestnajstiško… Kdor potrebuje več kot to, naj Googla.

1.3.1 Oblikovanje koles

Nazaj h kolesom. Večina izpisa je v redu, le deleže bi radi izpisali z manj decimalkami - recimo z eno samo. Dovolj je dodati `:.1f`.

[17]:
```
skupna = sum(razdalje.values())

for kolo, razdalja in razdalje.items():
    print(f"{kolo}: {razdalja} ({razdalja / skupna * 100:.1f} %)")
```

```
Canyon: 2932 (46.1 %)
Cube: 2939 (46.2 %)
Nakamura: 488 (7.7 %)
Stevens: 0 (0.0 %)
```

Celotna zgodba s kolesi je sicer nekoliko daljša. Podatke smo prebrali iz datoteke.

[18]:
```
voznje = {"Canyon": 0, "Cube": 0, "Nakamura": 0, "Stevens": 0}
razdalje = {"Canyon": 0, "Cube": 0, "Nakamura": 0, "Stevens": 0}
visine = {"Canyon": 0, "Cube": 0, "Nakamura": 0, "Stevens": 0}

for vrstica in open("kolesa.txt"):
    kolo, razdalja, visina = vrstica.split(",")
    voznje[kolo] += 1
    razdalje[kolo] += int(razdalja)
    visine[kolo] += int(visina)
```

V slovarjih `voznje`, `razdalje` in `visine` imamo število voženj ter skupno razdaljo in višino, prevoženo s posameznim kolesom. Kaj, ko bi to izpisali v obliki tabelice?

[19]:
```
sk_voz = sum(voznje.values())
sk_raz = sum(razdalje.values())
sk_vis = sum(visine.values())

for kolo in voznje:
    voz, raz, vis = voznje[kolo], razdalje[kolo], visine[kolo]
    print(f"{kolo:12}{voz:5} ({voz / sk_voz:>5.1%}) {raz:10} ({raz / sk_raz:>5.
 ↪1%}) {vis:10} ({vis / sk_vis:>5.1%}) ")
```

```
Canyon          26 (26.0%)       2766 (39.6%)      26392 (27.0%)
Cube            43 (43.0%)       3174 (45.4%)      66705 (68.3%)
Nakamura        22 (22.0%)        439 ( 6.3%)       1119 ( 1.1%)
Stevens          9 ( 9.0%)        607 ( 8.7%)       3395 ( 3.5%)
```

Novo je oblikovanje odstotkov: ne množimo s 100 in ne uporabimo `f` (recimo, zgoraj. `.1f`) temveč `%`. Konkretno, napisali smo `{voz / sk_voz:>5.1%}`. Količnik `voz / sk_voz` smo poravnali desno, izpisali na pet mest z eno decimalko in sicer kot odstotek. Ta pomeni, da bo Python reč kar sam pomnožil s `100` in še znak `%` bo dodal.

Mimogrede opazíte še, da smo spremenljivkam `voz`, `raz` in `vis` dali kratka imena, skupnemu številu voženj, dolžini in razdalji pa podobna, prav tako kratka imena. Že s tako kratkimi je vrstica kar dolga.

Izpis je videti že kar imenitno; še bolj imeniten bo, če mu dodamo imena stolpcev.

[20]:
```
sk_voz = sum(voznje.values())
sk_raz = sum(razdalje.values())
sk_vis = sum(visine.values())

print()
print(f"{'Kolo':6}{'Število voženj':>19}{'Razdalja':>19}{'Višina':>19}")
print("-" * (6 + 3 * 19))
for kolo in voznje:
    voz, raz, vis = voznje[kolo], razdalje[kolo], visine[kolo]
    print(f"{kolo:12}{voz:5} ({voz / sk_voz:>5.1%}) {raz:10} ({raz / sk_raz:>5.
 ↪1%}) {vis:10} ({vis / sk_vis:>5.1%})")
print("-" * (6 + 3 * 19))
```

```
Kolo       Število voženj           Razdalja             Višina
-----------------------------------------------------------------
Canyon          26 (26.0%)       2766 (39.6%)      26392 (27.0%)
Cube            43 (43.0%)       3174 (45.4%)      66705 (68.3%)
Nakamura        22 (22.0%)        439 ( 6.3%)       1119 ( 1.1%)
Stevens          9 ( 9.0%)        607 ( 8.7%)       3395 ( 3.5%)
-----------------------------------------------------------------
```

Tako lepo, da je že kar malo kičasto.

[ ]: