17. Nizi, Format

17.1. Primeri (nizi)

1. Sestavite funkcijo razstaviDatum(datum), ki prepozna datum oblike 'dd.mm.llll'. Vrne naj seznam s tremi celimi števili ali pa False, če datuma ne prepozna.
2. Sestavite funkcijo, ki preveri, ali je v nizu zapisan smiseln elektronski naslov. Naslov je smiseln, če je sestavljen iz več besed, kjer je vsaka beseda sestavljena iz črk angleške abecede, števk, podčrtajev in pomišljajev. Besede so ločene z afno ali s piko. Uporabimo lahko natanko eno afno. Za njo morata biti vsaj dve besedi.
3. Sestavite funkcijo, ki v danem nizu odstrani vse večkratne zaporedne pojavitve belih znakov (ostane samo en presledek). Odstrani naj tudi vse bele znake na začetku in na koncu niza.
4. Sestavite funkcijo, ki dano besedo zapiše olepšano, tako da bo vsaka črka v svojem okvirčku.
5. Sestavite funkcijo, ki dano besedo zapiše v okvirju. Z dodatnima parametroma naj bo določeno, koliko praznega prostora naj bo nad in pod besedo, ter koliko prostora naj bo levo in desno od besede.

17.2. Preoblikovanje nizov, metoda format

Z metodo s.format(...) si lahko pomagamo, kadar želimo sestaviti niz, ki bo poleg navadnih znakov vseboval vrednosti drugih izrazov. Mesto v nizu, kamor bi želeli vriniti vrednost izraza, posebej označimo z zavitimi oklepaji, v katere zapišemo zaporedno številko izraza (Parametre oštevilčimo od 0 naprej). Izraze napišemo kot parametre metodi. Primer: `'-{0}-{1}-'.format(8, 'abc')` vrne niz `'-8-abc-'`. V zavitih oklepajih lahko poleg številke izraza napišemo dvopičje in formatno določilo, ki določa, koliko znakov naj vrednost zavzema, kako naj bo poravnana, ... Več o formatnih določilih glej tukaj.

17.3.Pregled metode format.

Ta dokument precej natančno opiše možnosti formatiranja, ki jih ponuja metoda `format` na nizih. Spisan je predvsem z mislijo na tiste, ki že vejo, kaj ta metoda počne in kje je uporabna, se pa ne znajdejo med vsemi možnimi mini-ukazi formatu (ki jih ni malo).

Osnovna uporaba je

"blabla {...} bla bla {...} bla".format(argument1, argument2, ...)

Zaviti oklepaji na levi bodo po tem klicu zamenjani z argumenti na desni in metoda format vrne nov niz. Na kakšen način se bodo te zamenjave zgodile, določimo z vsebino zavitih oklepajev. V zavitih oklepajih ima vsebina obliko

{N:FPZS.DT},

oziroma

"bla bla {N:FPZS.D} bla".format(4, 3.13, "xxx", -37)

Pri tem velike črke le označujejo mesta določenih znakov oz. ukazov. In sicer (v drugačnem, bolj logičnem vrstnem redu kot zgoraj):

1) **N** - zaporedna številka argumenta od format, ki naj se vstavi

Primer: "{1}{0}{1}".format("Janez","Novak") --> "NovakJanezNovak"

2) **S** - najmanjše število znakov, t.j. minimalna dolžina končnega stringa. Če je argument daljši od S, se ne zgodi nič. Še je argument prekratek (ima premalo števk ali znakov), se ga podaljša do dolžine S. Običajno se podaljšuje z dodajanjem presledkov na desni, vendar glej tudi opis F in P spodaj.

Primer: Če uporabimo S=5 in je niz krajši, se popolni s presledki. "bla{0:5}bla".format("xyz") -->

"blaxyz bla" Če uporabimo S=6 in vstavimo število s 7 števkami, se ne zgodi nič.
"bla{0:6}bla".format(1234567) --> "bla1234567bla"

3) **F** - znak, s katerim se podaljša niz, da doseže zahtevano dolžino. Je smiseln samo v kombinaciji z S. Znak je lahko skoraj karkoli, vendar mora biti en sam. Privzeta vrednost je presledek F = " ".

Primer: Uporabili bomo F='_' in vstavili le 1. argument (argument 0 ostane neuporabljen). Hkrati zahtevam končno dolžino 5. "bla{1:_5}".format("krneki", 24) --> "___24" (TO NE DELUJE!)

4) **P** - poravnava -- določa, v katero smer se podaljšuje niz z znakom F, če S tako zahteva. Če namesto P pišemo znak ">", to pomeni desno poravnavo (dodajanje Fja na levi). Znak "<" pomeni levo poravnavo. Znak "^" pomeni sredinsko poravnavo. **Primeri**: "X{0:<5}X".format(12) --> "X12 X" # v naslednji vrstici bomo uporabili še F='!' "X{0:!>5}X".format("bla") --> "X!!blaX" "{0:-^10}".format("PYTHON") --> "--PYTHON--"

5) **Z** - izpis predznaka. Če namesto Z pišete znak "+", bo pred pozitivnimi števili python vstavil plus. Smiselno samo za zavite oklepaje, ki se bodo zamenjali s številom.

Primer: Uporabili bomo še F='0'. Druga vrstica kaže "normalno" obnašanje, brez uporabe Zja, in je samo za primerjavo. "imas {0:+05} evrov".format(12) --> "imas +0012 evrov" "imas {0:05} evrov".format(12) --> "imas 00012 evrov"

6) **D** - največje število decimalk, ki se izpišejo. Smiselno samo za zavite oklepaje, ki se bodo zamenjali z necelim številom. Če ima število manj kot D decimalk, se izpišejo vse. Če jih ima več, se število zaokroži na D decimalk. **POZOR:** prostor, ki ga porabijo te decimalke, se šteje v skupno dolzino (S).

Primer: "pi={0:5.3}".format(3.14159) --> "pi=3.1 " "pi={0:5.20}".format(3.14159) --> "pi=3.14159"

Vsi znaki/številke/ukazi namesto črk FPZSD so neobvezni, kateregakoli od njih lahko izpustite.

V tem primeru se upoštevajo **privzete vrednosti**:
- privzeta vrednost za F je presledek - privzeta vrednost za P je leva poravnava za nize, VENDAR desna poravnava za stevila (poskusite!)
- privzeta vrednost za Z je, da ne izpisuje plusa
- privzeta vrednost za S je 0 (in ker je vsak niz dolg vsaj 0 znakov, je učinek enak, kot da ne bi končne dolžine z ničemer določali)
- privzeta vrednost za D je neskončno. Če izpustimo vse znake za dvopičjem, lahko tudi dvopičje izpustimo. (Tak je primer pod točko 1 čisto zgoraj)

7) T - je tip predstavitve, če je argument število. Za cela števils pridejo v poštev "d", "x", "o", "b", "c" (decimlna, hexadecimalna, oktalna, binarna, znakovna predstavitev, za realna pa "f", "e", "%", ... (fiksna vejica, plavajoča vejica, procenti, ...) V resnici zna format narediti še nekaj drugih stvari, ampak so precej obskurne in poleg tega zahtevajo več znanja, kot ga trenutno imamo. Če koga vseeno zanima, velja link s predavanj:

http://docs.python.org/3.0/library/string.html#format-string-syntax

17.4. Formalna specifikacija:

```
replacement_field ::= "{" field_name ["!" conversion] [":"
format_spec] "}"
field_name ::= (identifier | integer) ("." attribute_name | "["
element_index "]")*
attribute_name ::= identifier
element_index ::= integer
conversion ::= "r" | "s" | "a"
format_spec ::= [[fill]align][sign][#][0][width][.precision][type]
fill ::= <a character other than '}'>
align ::= "<" | ">" | "=" | "^"
```

```
sign ::= "+" | "-" | " "
width ::= integer
precision ::= integer
type ::= "b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" |
"o" | "x" | "X" | "%"
```

17.5. Zgled:

Tabeliranje funkcije f(x) = 1/x**2.

```
print(" "+16*"_")

for i in range(1,11):

   print("|{0:5}|{1:10.5f}|".format(i,1/(i*i)))

print("|"+5*"_"+"|"+10*"_"+"|")
```

17.6. Naloge:

1. Napišite del programa, ki tabelira in pri tem lepo poravna števila od 0 do 99 v desetiški, oktalni in dvojiški obliki.

2. Napišite del programa, ki za dani n izpiše dvojiške n-razsežne vektorje, tako da tabelira naravna števila med 0 in 2**n-1 v dvojiškem sistemu. Npr. za n = 2 naj izpiše:

```
00

01

10

11
```