**Domača vaja 4**

Še nekaj nalog, ki smo si jih tudi izposodili iz predmeta Programiranje 1 z visokošolskega študija na FRI (pri prof. dr. J. Demšarju).

Vse funkcije shranite v datoteko vaje4.py in jih preverite s [testnim programom](http://ucilnica.fmf.uni-lj.si/pluginfile.php/16239/mod_page/content/7/test_vaje4.zip).

**Najdaljša beseda — najdaljsa(s)**

Napiši funkcijo najdaljsa, ki vrne najdaljšo besedo v nizu s.

>>> najdaljsa('an ban pet podgan')

'podgan'

def najdaljsa(s):

 naj = ''

 for beseda in s.split():

 if len(beseda) > len(naj):

 naj = beseda

 return naj

**Podobnost — podobnost(s1, s2)**

Napišite funkcijo, ki izračuna podobnost med dvema nizoma. Podobnost definirajmo kot število mest v katerih se niza ujemata.

sobota

robot

------

011110 -> 4

>>> podobnost('sobota', 'robot')

4

>>> podobnost('robot', 'sobota')

4

def podobnost(s1, s2):

 stevec = 0

 for i in range(min(len(s1), len(s2))):

 if s1[i] == s2[i]:

 stevec += 1

 return stevec

# Program lahko izboljšamo, če se spomnemo funkcije zip

def podobnost(s1, s2):

 stevec = 0

 for c1, c2 in zip(s1, s2):

 if c1 == c2:

 stevec += 1

 return stevec

# Ker se v Pythonu vrednosti True in False obnašata kot števili 0 in 1,

# bi lahko program zapisali tudi takole:

def podobnost(s1, s2):

 stevec = 0

 for c1, c2 in zip(s1, s2):

 stevec += c1 == c2

 return stevec

# Seveda se da tudi ta program skrajšati.

def podobnost(s1, s2):

 return sum(c1 == c2 for c1, c2 in zip(s1, s2))

**Sumljive besede — sumljive(s)**

Napiši funkcijo, ki vrne seznam vseh sumljivih besed v danem nizu. Beseda je sumljiva, če vsebuje tako črko u kot črko a.

>>> sumljive('Muha pa je rekla: "Tale juha se je pa res prilegla, najlepša huala," in odletela.')

['Muha', 'juha', 'huala,"']

def sumljive(s):

 sezSumljivih = []

 for beseda in s.split():

 if 'u' in beseda and 'a' in beseda:

 sezSumljivih.append(beseda)

 return sezSumljivih

**Vsi — vsi(xs)**

Napišite funkcijo vsi, ki sprejme seznam xs in vrne True, če so vse vrednosti v seznamu [resnične](http://docs.python.org/3.3/library/stdtypes.html#truth-value-testing). Elementi seznama xs so lahko poljubnega tipa, ne le bool.

>>> vsi([True, True, False])

False

>>> vsi([True, True])

True

>>> vsi([1, 2, 3, 0])

False

>>> vse(['foo', 42, True])

True

>>> vsi([])

True

def vsi(xs):

 for x in xs:

 if not x:

 return False

 return True

**Vsaj eden — vsaj\_eden(xs)**

Napišite funkcijo vsaj\_eden, ki deluje podobno kot vsi, le da vrne True, če je vsaj ena vrednost v seznamu resnična.

>>> vsaj\_eden([2, 3, 0])

True

>>> vsaj\_eden([])

False

def vsaj\_eden(xs):

 for x in xs:

 if x:

 return True

 return False

**Domine — domine(xs)**

Vrsta domin je podana s seznamom parov (terk), na primer [(3, 6), (6, 6), (6, 1), (1, 0)] ali [(3, 6), (6, 6), (2, 3)]. Napišite funkcijo, ki prejme takšen seznam in pove, ali so domine pravilno zložene. Za prvi seznam mora vrniti True in za drugega False.

def domine(domine):

 for i in range(len(domine) - 1):

 if domine[i][1] != domine[i + 1][0]:

 return False

 return True

**Vsota seznamov — vsota\_seznamov(xss)**

Podan je seznam seznamov, npr. [[2, 4, 1], [3, 1], [], [8, 2], [1, 1, 1, 1]]. Napiši funkcijo, ki v seznamu vrne vsote vseh elementov v posameznih podseznamih. Za gornji seznam naj funkcija vrne seznam [7, 4, 0, 10, 4], saj je, na primer, 2 + 4 + 1 = 7.

>>> vsota\_seznamov([[1, 1, 1], [1, 1]])

[3, 2]

>>> vsota\_seznamov([[2, 4, 1], [3, 1], [], [8, 2], [1, 1, 1, 1]])

[7, 4, 0, 10, 4]

def vsota\_seznamov(xss):

 vsote = []

 for xs in xss:

 vsota = 0

 for x in xs:

 vsota += x

 vsote.append(vsota)

 return vsote

# Ampak zakaj bi pisali na dolgo, če lahko napišemo tako:

def vsota\_seznamov(xss):

 return list(map(sum, xss))

**Največji podseznam — najvecji\_podseznam(xss)**

Podan je podoben seznam kot zgoraj. Napiši funkcijo, ki vrne podseznam z največjo vsoto elementov. Za gornji seznam mora funkcija vrniti [8, 2], saj je to podseznam z največjo vsoto, namreč 10.

>>> najvecji\_podseznam([[1, 1, 1], [1, 1]])

[1, 1, 1]

>>> najvecji\_podseznam([[2, 4, 1], [3, 1], [], [8, 2], [1, 1, 1, 1]])

[8, 2]

def najvecji\_podseznam(xss):

 najvecji = []

 najvecji\_vsota = float('-inf')

 for xs in xss:

 vsota = 0

 for x in xs:

 vsota += x

 if vsota > najvecji\_vsota:

 najvecji = xs

 najvecji\_vsota = vsota

 return najvecji

# Pomagamo si lahko tudi s funkcijo iz prejšnje naloge

def najvecji\_podseznam(xss):

 ys = vsota\_seznamov(xss)

 najvecji = float('-inf')

 for i, y in enumerate(ys):

 if y > ys[najvecji]:

 najvecji = i

 return xss[najvecji]

# \*Ali krajse:

def najvecji\_podseznam(xss):

 return max(xss, key=sum)

**Cezarjeva šifra — cezar(s)**

Napišite program, ki podan niz zašifrira z uporabo [cezarjeve šifre](http://sl.wikipedia.org/wiki/Cezarjeva_%C5%A1ifra). Preden se lotite naloge, se je morda vredno pozanimati kaj počneta funkciji [ord](http://docs.python.org/3.3/library/functions.html#ord) in [chr](http://docs.python.org/3.3/library/functions.html#chr). Predpostavite lahko, da niz s vsebuje le male črke angleške besede in presledke.

>>> cezar('the quick brown fox jumps over the lazy dog')

'wkh txlfn eurzq ira mxpsv ryhu wkh odcb grj'

def cezar(s):

 cipher = ''

 for c in s:

 if 'a' <= c <= 'w':

 cipher += chr(ord(c) + 3)

 elif 'x' <= c <= 'z':

 cipher += chr(ord(c) - 23)

 else:

 cipher += c

 return cipher

Nalogo cezar lahko rešimo tudi s parom funkcij maketrans in translate.

def cezar(s):

 alphabet = 'abcdefghijklmnopqrstuvwxyz'

 trans = str.maketrans(alphabet, alphabet[3:] + alphabet[:3])

 return s.translate(trans)

**Multiplikativni range — mrange(start, faktor, dolzina)**

Napišite funkcijo, ki vrne seznam, kjer je vsako naslednje število za faktor večje od prejšnjega. Npr., v seznamu [1, 2, 4, 8, 16] je vsako naslednje število 2-krat večje od prejšnjega.

>>> print(mrange(7, 4, 7))

[7, 28, 112, 448, 1792, 7168, 28672]

def mrange(s, r, l):

 xs = []

 for i in range(l):

 xs.append(s)

 s \*= r

 return xs

**Slikaj — slikaj(f, xs)**

Napišite funkcijo slikaj, ki sprejme funkcijo f in seznam [x\_1, x\_2, ..., x\_n] in vrne nov seznam [f(x\_1), f(x\_2), ..., f(x\_n)].

>>> slikaj(abs, [-5, 8, -3, -1, 3])

[5, 8, 3, 1, 3]

>>> slikaj(len, "Daydream delusion limousine eyelash".split())

[8, 8, 9, 7]

def slikaj(f, xs):

 ys = []

 for x in xs:

 ys.append(f(x))

 return ys