Ponovitev

Branje datotek

Racunanje povprečne vrednosti

In [1]:
```
f = open("../../datoteke/teze.txt")
vsota = 0
prsti = 0
for vrstica in f:
    vsota = vsota + int(vrstica)
    prsti = prsti + 1
print(vsota / prsti)
```

```
70.5
```

In [2]:
```
izhodna_datoteka = open("./povprecje.txt", "w", encoding="utf8")
```

In [3]:
```
izhodna_datoteka.write(str(vsota / prsti) + "\n")
```

Out[3]: 5

In [4]:
```
izhodna_datoteka.write("to je povprečje\n\n\n")
```

Out[4]: 18

In [5]:
```
izhodna_datoteka.write("to je konec.\n")
```

Out[5]: 13

In [6]:
```
izhodna_datoteka.close()
```

In [7]:
```
izhodna_datoteka.write("še tole sem pozabil dodati")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-7-85e60acdd987> in <module>
----> 1 izhodna_datoteka.write("še tole sem pozabil dodati")

ValueError: I/O operation on closed file.
```

Računanje BMI

Beri iz datoteke osebe in podatke BMI shrani v novo datoteko `bmi.txt`.

In [8]:
```
f = open("bmi.txt", "w")

g = open("../../datoteke/osebe.txt", encoding="utf8")
for vrstica in g:
    ime, teza, visina = vrstica.split(" ")
    bmi = int(teza) / float(visina) ** 2

    #f.write(ime + " " + str(bmi) + "\n")
    print(f"{ime:20}{bmi:8.2f}")
    f.write(f"{ime:20}{bmi:8.2f}\n")

f.close()
```

```
Ana                  24.91
Berta                25.11
Cilka                25.71
Dani                 25.31
Eva                  22.22
Fanči                23.51
```

Delo z nizi

In [9]:
```
s = "Ana 72 1.70"
```

In [10]:
```
s.count(" ")
```

Out[10]: 2

In [11]:
```
sequence = "ACGATCAGCACGGGACAGCGACGGGGGCAGCAGGGTGCACG"
```

In [12]:
```
sequence.count("A")
```

Out[12]: 10

In [13]:
```
sequence.count("ACG")
```

Out[13]: 4

In [14]:
```
sequence.lower()
```

Out[14]: 'acgatcagcacgggacagcgacgggggcagcagggtgcacg'

In [15]:
```
sequence.replace("T", "Ψ")
```

Out[15]: 'ACGAΨCAGCACGGGACAGCGACGGGGGCAGCAGGGΨGCACG'

In [16]:
```
sequence
```

Out[16]: 'ACGATCAGCACGGGACAGCGACGGGGGCAGCAGGGTGCACG'

In [17]:
```
"    asgfa a  iji    ".strip()
```

Out[17]: 'asgfa a iji'

Seznami

In [18]:
```
imena = ["Ana", "Berta", "Cilka", "Dani", "Ema", "Fanči"]
```

In [19]:
```
drugi = imena[2]
print(drugi)
```

```
Cilka
```

In [20]:
```
len(imena)
```

Out[20]: 6

In [21]:
```
imena[0]
```

Out[21]: 'Ana'

In [22]:
```
imena[-1]
```

Out[22]: 'Fanči'

In [23]:
```
imena[2:5]
```

Out[23]: ['Cilka', 'Dani', 'Ema']

In [24]:
```
imena[:-2]
```

Out[24]: ['Ana', 'Berta', 'Cilka', 'Dani']

In [25]:
```
teze = [56, 76, 80, 67, 60]
```

In [26]:
```
vsota = 0
for teza in teze:
    vsota = vsota + teza
print(vsota / len(teze))
```

```
67.8
```

In [27]:
```
teze.sort()
```

In [28]:
```
teze
```

Out[28]: [56, 60, 67, 76, 80]

In [29]:
```
teze.append(92)
```

In [30]:
```
teze
```

Out[30]: [56, 60, 67, 76, 80, 92]

In [31]:
```
max(teze)
```

Out[31]: 92

In [32]:
```
min(teze)
```

Out[32]: 56

branje sekvenc DNA

In [33]:
```
for vrstica in open("../../datoteke/qwerty-dna.txt"):
    gen, sekvenca = vrstica.split()
    print(gen)
    print(sekvenca)
```

```
ASDF13
gcaactgttggacggctacagtgacggttggtagaactgagtcggtttaaggactcacacatcgcgggtctgca
aagtgtaatctacaagggagcccgag
SDFG14
cgaagggcaatcggaagttgaggttcgtcatattaagtttggggaacgccgacatctaaatcttttaggtgata
aatgcctaaatcagattcaatgtatt
DFGH15
cgacctcgtaaaatgacaaacactgtcgtggagcagtattcggtcatgccgcccgagccctaccaatcgagttc
aactatcgctaactcgcgatgagcct
FGHJ16
tcgcgggtagcccacagccgggcctgattacagaggggtgaattcgatgcttgatgcggattcctggtaagctc
cgccgtgcgaccgacaactctcgact
GHJK17
tggtgatgtggtacatctttgaaaggctcaccgtgaacaaaagtgtattacaatcaacgagccccagggactga
tccctcaacaagggcacccaagaagt
HJKL18
tacagacactatcgctcccgtagctggaggatttcacatgatctaagcaaagccgtagtgggagttcctatggc
aataagcgaccttctataaccgagag
ZXCV19
tcatgcatgttaggttacatctaggctatgcctgtcccagtcagcaggtgggcctagattaagaaatgcccggt
taggcaacacaacaccggtgtccttt
XCVB20
cgctacgtatgtccctaatcaagggctcatggtgctagccagggtcggggctagtttttaaggtatttctgccc
ccaacaaggagccagataggcccctt
CVBN21
acttggccactttactcctgcaatctttagtcctgggggagtttaaaatcattccagctgggatgggtctcta
tcctctccctgtaataacaacaaacg
VBNM22
tggtctttaagattaactgctcattaggatctgtctccaaacactgttaccgccggcaatcacaggagaatcag
tcacctaagttgcgtaggccatatcc
```

Danes

Vaja: poišči največjo maso

Pomagate si lahko s to psevdokodo:

```
for each line in the file:
    if the number in this line is greater than the biggest
so far:
        the biggest so far is the number in this line

print the biggest so far
```

In [34]:
```
m = 0
for line in open("../../datoteke/teze.txt"):
    if float(line) > m:
        m = float(line)
print(m)
```

```
85.0
```

alternativa, vse preberi v seznam in potem poišči največji element

In [35]:
```
mase = []
for line in open("../../datoteke/teze.txt"):
    mase.append(float(line))
print(max(mase))
```

```
85.0
```

Še ena vaja:

Zadnjič smo izpisali tri gene, ki imajo največ ponovitev `A':

In [36]:
```
geni = []

for vrstica in open("../../datoteke/qwerty-dna.txt"):
    gen, sekvenca = vrstica.split()
    geni.append((sekvenca.count("a"), gen))

geni.sort(reverse=True)
geni = geni[:3]
print(geni)
print(len(geni))
for par in geni:
    pojavitve, gen = par
    print(gen, pojavitve)
```

```
[(32, 'GHJK17'), (31, 'SDFG14'), (29, 'HJKL18')]
3
GHJK17 32
SDFG14 31
HJKL18 29
```

Napišite program, ki bo izpisal samo tisti gen, z največ `A` ne da bi si zapomnil vse dolžine.

In [37]:
```
max_as = 0
max_name = ""
for line in open("../../datoteke/qwerty-dna.txt"):
    name, sequence = line.split()
    seq_a = sequence.count("a")
    print(name, seq_a)
    if seq_a > max_as:
        max_as = seq_a
        max_name = name

print("-" * 9)
print(max_name, max_as)
```

```
ASDF13 26
SDFG14 31
DFGH15 26
FGHJ16 19
GHJK17 32
HJKL18 29
ZXCV19 24
XCVB20 21
CVBN21 24
VBNM22 27
---------
GHJK17 32
```

Slovar

In [38]:
```
teze = {'Ana': 72, 'Berta': 85, 'Eva': 50}
```

In [39]:
```
teze['Ana']
```

Out[39]:
```
72
```

In [40]:
```
teze['Eva']
```

Out[40]:
```
50
```

In [41]:
```
teze['Martin']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-41-dad95dc2d519> in <module>
----> 1 teze['Martin']

KeyError: 'Martin'
```

Določimo lahko vrednost, ki naj se uporabi v primeru, da ključ ne obstaja.

In [42]:
```
teze.get('Martin', 0)
```

Out[42]:
```
0
```

dodajanje elementov

In [43]:
```
teze['Cilka'] = 70
```

In [44]:
```
teze
```

Out[44]:
```
{'Ana': 72, 'Berta': 85, 'Eva': 50, 'Cilka': 70}
```

In [45]:
```
teze['Fanči'] = 64
```

In [46]:
```
teze
```

Out[46]:
```
{'Ana': 72, 'Berta': 85, 'Eva': 50, 'Cilka': 70, 'Fanči': 64}
```

odstranjevanje

In [47]:
```
del teze['Ana']
```

In [48]:
```
teze
```

Out[48]:
```
{'Berta': 85, 'Eva': 50, 'Cilka': 70, 'Fanči': 64}
```

In [49]:
```
del teze['Ana']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-49-61cd90e9374f> in <module>
----> 1 del teze['Ana']

KeyError: 'Ana'
```

In [50]:
```
teze
```

Out[50]:
```
{'Berta': 85, 'Eva': 50, 'Cilka': 70, 'Fanči': 64}
```

dostopanje do vseh elementov

In [51]:
```
teze.items()
```

Out[51]:
```
dict_items([('Berta', 85), ('Eva', 50), ('Cilka', 70), ('Fanči', 64)])
```

In [52]:
```
for ime, teza in teze.items():
    print(ime, teza)
```

```
Berta 85
Eva 50
Cilka 70
Fanči 64
```

samo ključi

In [53]:
```
teze.keys()
```

Out[53]:
```
dict_keys(['Berta', 'Eva', 'Cilka', 'Fanči'])
```

samo vrednosti

In [54]:
```
teze.values()
```

Out[54]:
```
dict_values([85, 50, 70, 64])
```

Slovar lahko vsebuje tudi sezname.

Recimo, seznam zgodovine spreminjanja telesne mase. Zgodovine so različno dolge.

In [55]:
```
teze_zgodovina = {'Ana': [70, 71, 72], 'Berta': [90, 85], 'Cilka': [77,
```

In [56]:
```
teze_zgodovina
```

Out[56]:
```
{'Ana': [70, 71, 72],
 'Berta': [90, 85],
 'Cilka': [77, 75, 72, 70],
 'Eva': [50, 48, 50]}
```

Vaja: napiši, kdo je shujšal, kdo pa se je zredil

In [57]:
```
for ime, teze in teze_zgodovina.items():
    if teze[0] > teze[-1]:
        print(f"{ime} je hujšal(a) iz {teze[0]} na {teze[-1]}")
    elif teze[0] < teze[-1]:
        print(f"{ime} se je zredil(a) iz {teze[0]} na {teze[-1]}")
    else:
        print(f"{ime} ostaja enake mase")
```

```
Ana se je zredil(a) iz 70 na 72
Berta je hujšal(a) iz 90 na 85
Cilka je hujšal(a) iz 77 na 70
Eva ostaja enake mase
```

Vaja: napiši program, ki izpiše vse telefonske posamezne osebe

Telefonske so zapisane v `datoteke/telefonske.txt`.

In [58]:
```
imenik = {}
for line in open("../../datoteke/telefonske.txt"):
    ime, telefonska = line.split()
    imenik.setdefault(ime, []).append(telefonska)
imenik
```

Out[58]:
```
{'Ana': ['0409381326', '0413339231'],
 'Berta': ['0412399483'],
 'Cilka': ['0312791485', '0417721128', '0407721128'],
 'Dani': ['23013905'],
 'Luka': ['0312921789']}
```

Vaja: napiši program, ki izpiše število telefonskih, ki jih ima posamezna oseba

In [59]:
```
imenik = {}
for line in open("../../datoteke/telefonske.txt"):
    ime, telefonska = line.split()
    imenik.setdefault(ime, []).append(telefonska)

for ime, telefonske in imenik.items():
    print(f"{ime} ima {len(telefonske)} telefonsko(ih) številk(o)")
```

```
Ana ima 2 telefonsko(ih) številk(o)
Berta ima 1 telefonsko(ih) številk(o)
Cilka ima 3 telefonsko(ih) številk(o)
Dani ima 1 telefonsko(ih) številk(o)
Luka ima 1 telefonsko(ih) številk(o)
```

branje terk v slovar

In [60]:
```
seq = "ACGATCAGCACGGGACAGCGACGGGGGCAGCAGGGTGCACG"
```

In [61]:
```
frek = {}
for c in seq:
    frek[c] = frek.get(c, 0) + 1
print(frek)
```

```
{'A': 10, 'C': 11, 'G': 18, 'T': 2}
```

definiranje funkcij

In [62]:
```
def prestej(s):
    frek = {}
    for c in s:
        frek[c] = frek.get(c, 0) + 1
    return frek
```

In [63]:
```
prestej("ACGATCAGCACGGGAC")
```

Out[63]:
```
{'A': 5, 'C': 5, 'G': 5, 'T': 1}
```

In [64]:
```
prestej("Janko in Metka")
```

Out[64]:
```
{'J': 1,
 'a': 2,
 'n': 2,
 'k': 2,
 'o': 1,
 ' ': 2,
 'i': 1,
 'M': 1,
 'e': 1,
 't': 1}
```

spremenljivke so lokalne

In [65]:
```
a=4

def t(a):
    a=2
    print('med klicem:', a)

print('pred klicem:', a)
t("eee")
print('po klicu:', a)
```

```
pred klicem: 4
med klicem: 2
po klicu: 4
```

Štetje terk.

In [66]:
```
niz = "012345678901"
```

neprekrivajoče

In [67]:
```
for c in niz[::3]:
    print(c)
```

```
0
3
6
9
```

prekrivajoče

In [68]:
```
i = 0
counts = {}
while i < len(niz)-2:
    triplet = niz[i : i + 3]
    if triplet not in counts:
        counts[triplet] = 0
    counts[triplet] += 1
    i += 1
counts
```

Out[68]:
```
{'012': 1,
 '123': 1,
 '234': 1,
 '345': 1,
 '456': 1,
 '567': 1,
 '678': 1,
 '789': 1,
 '890': 1,
 '901': 1}
```

Vaja: napiši funkcijo, ki prešteje vse prekrivajoče trojke

In [69]:
```
def triplet_counts(seq):
    i = 0
    counts = {}
    while i < len(seq)-2:
        triplet = seq[i : i + 3]
        if triplet not in counts:
            counts[triplet] = 0
        counts[triplet] += 1
        i += 1
    return counts
```

In [70]:
```
triplet_counts(niz)
```

Out[70]:
```
{'012': 1,
 '123': 1,
 '234': 1,
 '345': 1,
 '456': 1,
 '567': 1,
 '678': 1,
 '789': 1,
 '890': 1,
 '901': 1}
```

Vaja: napiši funkcijo, ki prešteje vse prekrivajoče terke poljubne dolžine

In [71]:
```
def kmer_count(seq, k):
    i = 0
    counts = {}
    while i < len(seq)-(k-1):
        triplet = seq[i : i + k]
        if triplet not in counts:
            counts[triplet] = 0
        counts[triplet] += 1
        i += 1
    return counts
```

In [72]:
```
kmer_count(niz, 2)
```

Out[72]:
```
{'01': 2,
 '12': 1,
 '23': 1,
 '34': 1,
 '45': 1,
 '56': 1,
 '67': 1,
 '78': 1,
 '89': 1,
 '90': 1}
```

In [73]:
```
kmer_count(niz, 5)
```

Out[73]:
```
{'01234': 1,
 '12345': 1,
 '23456': 1,
 '34567': 1,
 '45678': 1,
 '56789': 1,
 '67890': 1,
 '78901': 1}
```

Vaja: napiši program, ki za dano datoteko FASTA izpiše gen z največ ponovitvami, določenega zaporedja.

In [74]:
```
zap = "agt"

naj_frek = 0
naj_gen = '?'

for line in open("../../datoteke/qwerty-dna.txt"):
    gen, sekvenca = line.split()
    kmers = kmer_count(sekvenca, 3)
    frek = kmers.get(zap, 0)

    if frek > naj_frek:
        naj_frek = frek
        naj_gen = gen
print(f"Zaporedje '{zap}' se pojavi največkrat ({naj_frek}-krat) v genu {
```

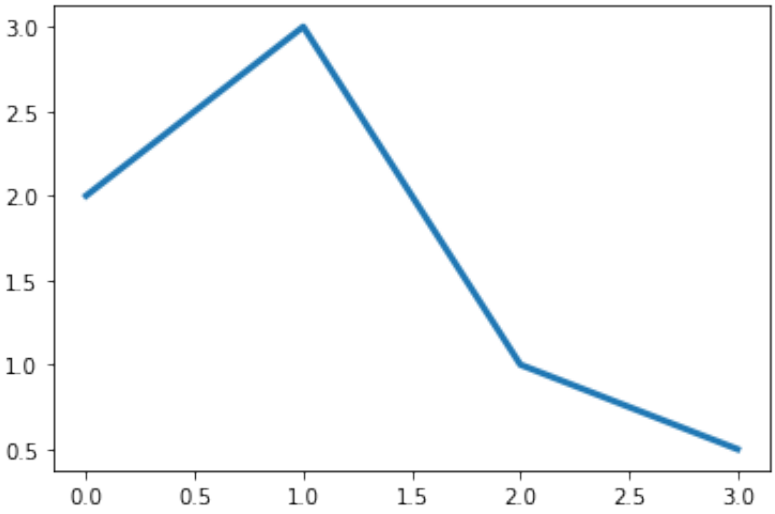
```
Zaporedje 'agt' se pojavi največkrat (3-krat) v genu ASDF13
```

Risanje grafov

In [75]:
```
import matplotlib.pyplot as plt
```

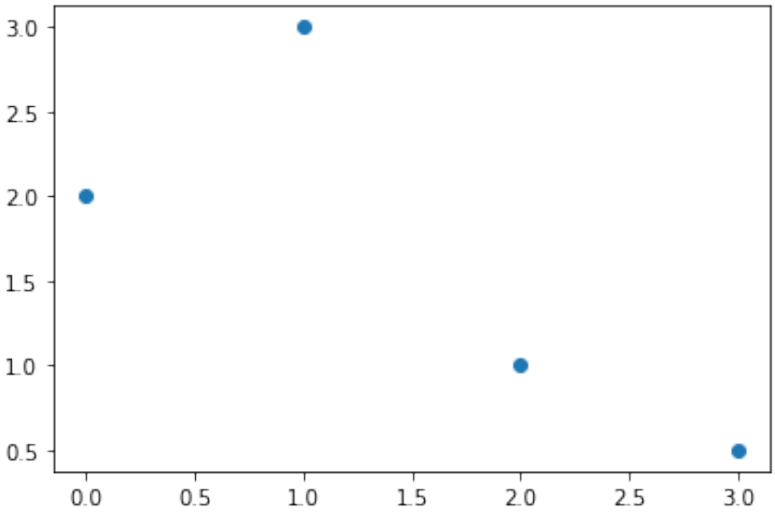
In [76]:
```
plt.plot([2, 3, 1, 0.5], lw=3)
```

Out[76]:
```
[<matplotlib.lines.Line2D at 0x7f767ee39bb0>]
```



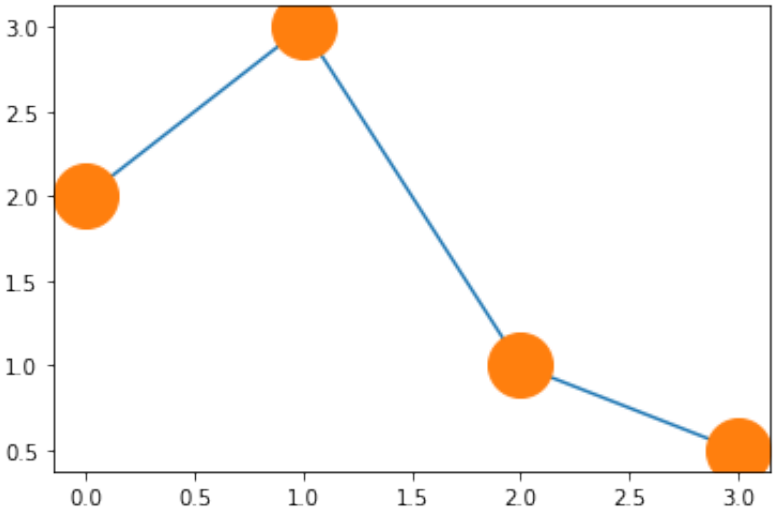
In [77]:
```
plt.plot([2, 3, 1, 0.5], 'o', )
```

Out[77]:
```
[<matplotlib.lines.Line2D at 0x7f7676d50760>]
```



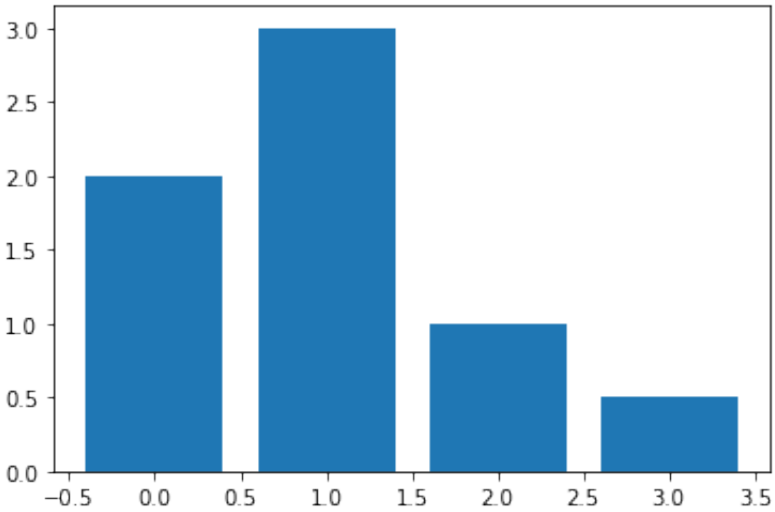
In [78]:
```
plt.plot([2, 3, 1, 0.5])
plt.plot([2, 3, 1, 0.5], 'o', ms=30)
```

Out[78]:
```
[<matplotlib.lines.Line2D at 0x7f7676cbcd60>]
```

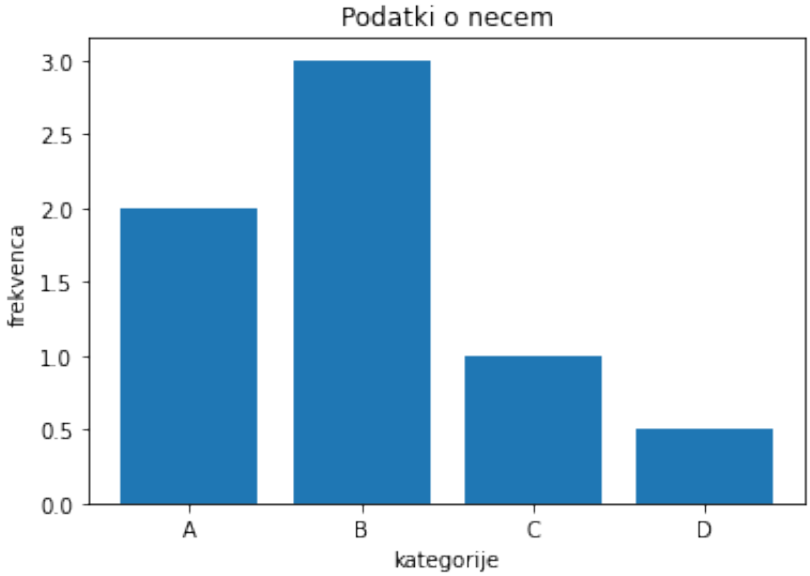


In [79]:
```
plt.bar([0,1,2,3], [2, 3, 1, 0.5])
```

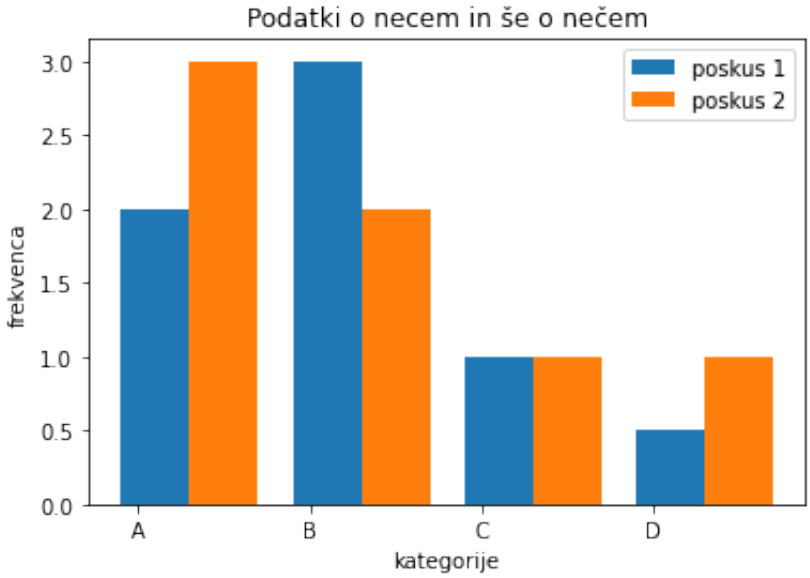
Out[79]: <BarContainer object of 4 artists>



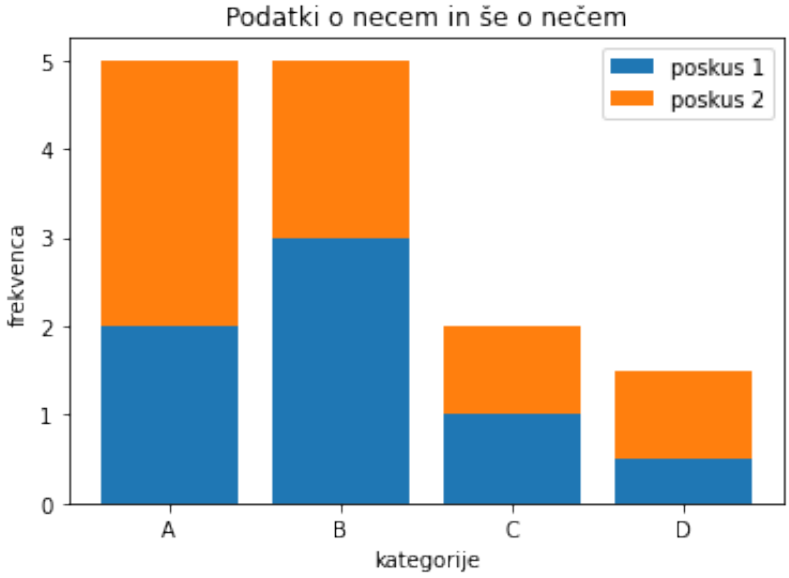
In [80]:
```
plt.bar([0,1,2,3], [2, 3, 1, 0.5])
plt.title('Podatki o necem')
plt.xlabel('kategorije')
plt.ylabel('frekvenca')
plt.xticks([0,1,2,3], ['A', 'B', 'C', 'D'])
plt.savefig('graf.png')
```



In [81]:
```
plt.bar([0.1,1.1,2.1,3.1], [2, 3, 1, 0.5], width=0.4, label='poskus 1')
plt.bar([0.5,1.5,2.5,3.5], [3, 2, 1, 1], width=0.4, label='poskus 2')
plt.title('Podatki o necem in še o nečem')
plt.xlabel('kategorije')
plt.ylabel('frekvenca')
plt.xticks([0,1,2,3], ['A', 'B', 'C', 'D'])
plt.legend()
plt.savefig('graf.png')
```



In [82]:
```
plt.bar([0,1,2,3], [2, 3, 1, 0.5], label='poskus 1')
plt.bar([0,1,2,3], [3, 2, 1, 1], bottom=[2, 3, 1, 0.5], label='poskus 2')
plt.title('Podatki o necem in še o nečem')
plt.xlabel('kategorije')
plt.ylabel('frekvenca')
plt.xticks([0,1,2,3], ['A', 'B', 'C', 'D'])
plt.legend()
plt.savefig('graf.png')
```

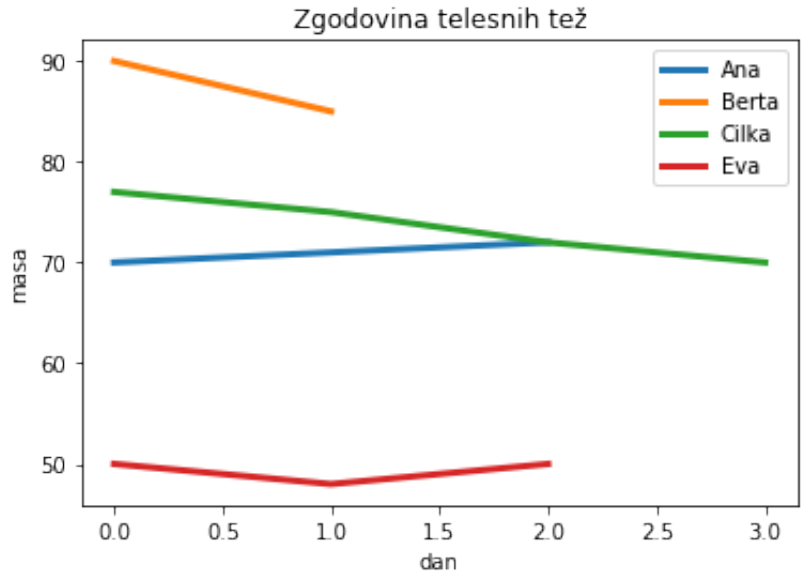


Vaja: nariši, kako so se spreminjale teže

In [83]:
```
teze_zgodovina = {'Ana': [70, 71, 72], 'Berta': [90, 85], 'Cilka': [77,
```

In [84]:
```
plt.title('Zgodovina telesnih tež')
for ime, teze in teze_zgodovina.items():
    plt.plot(range(len(teze)), teze, label=ime, lw=3)
plt.xlabel('dan')
plt.ylabel('masa')
plt.legend()
```

Out[84]:
```
<matplotlib.legend.Legend at 0x7f7676aecdc0>
```



Risanje grafa porazdelitve A+T%

In [85]:
```
import random
random.seed(42)
niz = "".join([random.choice(["A", "T", "C", "G"]) for x in range(500)])
```

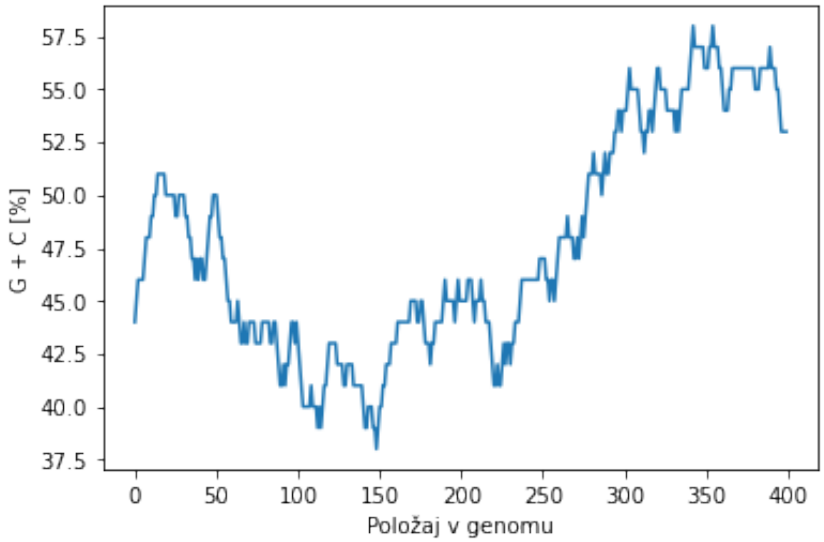
In [86]:
```
niz
```

Out[86]:
```
'AACTTTAAGAAATTATGTGCATGCCTTCAAGACCCAGAGACCTAATCATAGCGCTCCTCATTTGGCTCATA
CGCATCTGGGTCTTCGGCTTGAAATTGAGGGCAACCACGTGACTACTTCTACGAACCTATAAGATTGTCGTT
CGCGGATTACATTAAATAACATCGTTGTGGTAAGCGGGAAAGCATTTGTGTCGTAGAAAATTGGGTGATGAG
CGCGGTTCTAACAAGTAATAATGATAAGCCTCTCGTCGCAAGAATCTCATCCTGCACATCAATCCTCTCGCA
AGCAACTCTGGAAATACTGTACCACTTACGTTTTGATCGTCTAGAGTTGCCTTATGCCACCGCAACTCAAGC
CGAGTCAGATCGACCACCGCGCTTGGTCGACCTGCGGGTGTACCATCTTTAATGAGGTGGGTTAAGTTGATA
GTGCGGGTGGCTCGTCGACTCCCATTTGTTAGGCGGATGGAGGACCGGTGTCGGAGCGTGTAGAAGTGT'
```

In [87]:
```
def gc(sek):
    return sek.count("G") + sek.count("C")

vrednosti = []
for x in range(len(niz)-100):
    v = gc(niz[x:x+100])
    vrednosti.append(v)
```

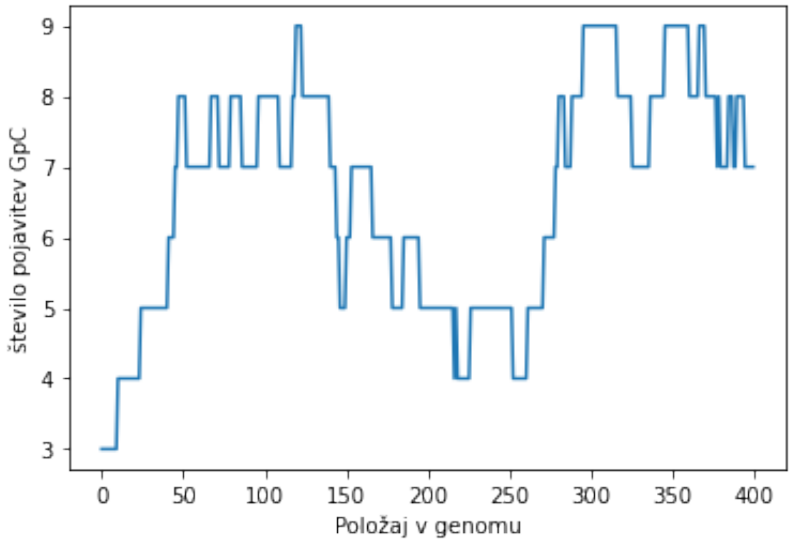
In [88]:
```
plt.plot(vrednosti)
plt.xlabel('Položaj v genomu')
plt.ylabel('G + C [%]');
```



In [89]:
```
def CpG(sek):
    return sek.count("CG")

vrednosti = []
for x in range(len(niz)-100):
    v = CpG(niz[x:x+100])
    vrednosti.append(v)
```

In [90]:
```
plt.plot(vrednosti)
plt.xlabel('Položaj v genomu')
plt.ylabel('število pojavitev GpC');
```


število pojavitev GpC
Položaj v genomu
3
4
5
6
7
8
9
0
50
100
150
200
250
300
350
400

BioPython