Príloha A

k záverečnej práci ***Vyučovanie programovania v jazyku Python***

Mgr. Martin Janček, PF KU 2018

Zbierka riešených úloh v jazyku Python

Príloha obsahuje zbierku riešených úloh v jazyku Python vytvorených podľa Cieľových požiadaviek na vedomosti a zručnosti maturantov z informatiky (CP) platných od školského roku 2018/2019. Nadpis úlohy obsahuje kompetenciu z CP, zadanie úlohy, príklad riešenia a tipy na riešenie.

Elektronická verzia zbierky je dostupná na webovej adrese **http://python.mateusko.sk**.

1) 3.3a Zistiť, či číslo spĺňa zadané kritérium (1)

Zadanie:

- Vytvorte program, ktorý zistí, či dané číslo končí na cifru 8 a zároveň je deliteľné troma.

```
x = int(input("Zadaj cislo: "))

# je číslo deliteľné troma?
p1 = x % 3 == 0

# má číslo na konci osmičku?
p2 = x % 10 == 8

if p1 and p2:
    print("Číslo spĺňa podmienku")
else:
    print("Číslo nespĺňa podmienku")
```

Tip na riešenie:

- Využite operátor % - zvyšok po celočíselnom delení. Ak je číslo n deliteľné číslom d, tak zvyšok po delení d je nula $n \% d = 0$
  - Zložitejšie podmienky môžete vyhodnotiť postupne. Medzi výsledok logickej operácie môžete uložiť do premennej (p1, p2) a ďalej s ním pracovať.

2) 3.3a Zistiť, či číslo spĺňa zadané kritérium (2)

Zadanie:

- Vytvorte program, ktorý overí, či zadané rodné číslo je platné. Pozn. desaťmiestne rodné číslo musí byť deliteľné číslom 11 a 3. - 4. cifra musí vyjadrovať poradové číslo mesiaca 01 -január. U žien je k poradovému číslu mesiaca pripočítané číslo 50. (Náročnejší variant: skontrolujte aj číslo dňa v mesiaci – pozor na priestupné a nepriestupné roky)

```
s = input("Zadaj desaťmiestne rodné číslo:")
x = int(s)

# je deliteľné 11-mi?
p1 = x % 11 == 0

m = int(s[2:4])

if m > 50:
    m = m - 50

if m > 1 and m <= 12 and len(s) == 10 and p1:
    print("Rodné číslo má správny tvar.")
else:
    print("Rodné číslo má nesprávny tvar.")
```

Tip na riešenie:

- Rodné číslo by malo mať 10 znakov. Dĺžka reťazca za dá zistiť funkciou `len(reťazec)`
- RČ je deliteľné jedenástkou, čiže zvyšok po delení 11 je nula `x % 11 == 0`. Operátor % pracuje len s celými číslami, nie s reťazcami. Najprv konvertuje reťazec na celé číslo funkciou `int(reťazec)`
- Vybrať podreťazec z reťazca možno zápisom `reťazec[a:b]`. `a, b` sú indexy, ktoré určujú od ktorého po ktorý znak sa vyberie podreťazec z pôvodného reťazca.
- Ak sa vám javí riešenie ako neúplné, vylepšite ho o kontrolu dňa z 5.-6. číslice RČ. Máme mesiace s rôznym počtom dní a aj priestupné roky!

3) 3.3b *Realizovať výpočty s desatinnými číslami (1)*

Zadanie:

- Vytvorte program na prevod stupňov Celzia na Kelviny a stupne Fahrenheita. Pozn. Kelviny = Celziove_stupne + 273,15; Fahrenheitove_stupne = 1,8 . Celziove_stupne + 32

```
c = float(input("Zadaj teplotu v C: "))

k = c + 273.15
f = 1.8 * c + 32

print("Teplota v K je", k)
print("Teplota v F je", f)
```

Tip na riešenie:

- Keď potrebujete zmeniť reťezec obsahujúci vyjadrenie desatinného čísla na typ desatinné číslo, použite funkciu `float(reťazec)`

4) 3.3b Realizovať výpočty s desatinnými číslami (2)

Zadanie:

- ❖ Vytvorte program, ktorý vyrieši lineárnu rovnicu $a.x + b = c$. Program vyžiada vstupné hodnoty a, b, c a vypíše výsledok. Pozn. ak $a = 0$ môžu nastať dva prípady 1) $b = c$ rovnica má nekonečne veľa riešení 2) $b \neq c$ rovnica nemá riešenie.

```
print(" Riešenie rovnice a.x + b = c ")
a = float(input("Zadaj a: "))
b = float(input("Zadaj b: "))
c = float(input("Zadaj c: "))

if a != 0:
    print("x = ", (c - b) / a)
else:
    if b == c:
        print("Rovnica má nekonečne veľa riešení.")
    else:
        print("Rovnica nemá riešenie")
```

Tip na riešenie:

- ☞ Návod pre tých, ktorí sa nekamarátia s matematikou: $x = (c - b) / a$

5) 3.3c Previesť čísla medzi číselnými sústavami (1)

Zadanie:

- ❖ Vytvorte program na prepočet čísla z desiatkovej do dvojkovej sústavy.

```
d = int(input("Zadaj celé kladné číslo v desiatkovej sústave: "))

s = ""

while d > 0:
    s = str(d % 2) + s
    d = d // 2

print("Číslo v dvojkovej sústave je ", s)
```

Tip na riešenie:

- ☞ Algoritmus premeny čísla d z desiatkovej do dvojkovej sústavy:
  1. Zvyšok po delení čísla dvojkou je dvojková číslica (0 alebo 1), ktorú zapíšeme do reťazca **s** – to bude premenené číslo v dvojkovej sústave.
  2. Výsledok celočíselného delenia (operátor //) čísla d dvojkou zapíšeme do d.
  3. Tento postup opakujeme dovtedy, kým je d > 0. Ďalšie vypočítané zvyšky po delení dvojkou postupne pridávame na začiatok reťazca **s**.
- ☞ Príklad: 10 : 2 = 5 zvyšok 0 … zapíšeme **0**
  5 : 2 = 2 zvyšok 1 … zapíšeme **1**0
  2 : 2 = 1 zvyšok 0 … zapíšeme **0**10
  1 : 2 = 0 zvyšok 1 … zapíšeme **1**010 ← výsledok

6) 3.3c Previesť čísla medzi číselnými sústavami (2)

Zadanie:

- Vytvorte program na prepočet čísla zo 16-ovej do desiatkovej sústavy

```
hx = input("Zadaj číslo v šestnástkovej sústave: ")

dobre = True # ak je v zadanom reťazci chyba = false
y = 0        # výsledok - číslo v desiatkovej sústave

for c in hx:

    #určenie desiatkovej hodnoty hexadecimalneho symbolu
    if c >= '0' and c <= '9':
        x = int(c)
    elif c >= 'a' and c <= 'f':
        x = ord(c) - ord('a') + 10
    elif c >= 'A' and c <= 'F':
        x = ord(c) - ord('A') + 10
    else:
        dobre = False # našiel sa nepovolený znak

    if dobre:
        y = y * 16 + x
    else:
        break

if dobre:
    print("Číslo v desiatkovej sústave je ", y)
else:
    print("Zadané hexadecimálne číslo obsahuje nepovolený znak")
```

Tip na riešenie:

- Šestnásková sústava obsahuje okrem číslic 0-9 písmená A-F s významom A=10, B=11, C=12, D=13, E=14, F=15.
- Ak chceme znak ‘A‘ zmeniť na číslo 10, môžeme použiť funkciu `ord(znak)`, ktorý vráti poradové číslo (kód) znaku v tabuľke znakov. V tejto tabuľke sú znaky A-Z zoradené tesne za sebou, podobne tiež znaky a-z. Výraz `ord(znak)-ord('A')` vráti poradové číslo znaku počítané od ‘A‘ (A=0, B=1, C=3…)
- Algoritmus premeny čísla v šestnástkovej sústave do desiatkovej sústavy:
    1. Premennú **y**, v ktorej bude premenené desiatkové číslo vynulujeme.
    2. Vezmeme prvý znak šestnástkového čísla zľava, ak je to písmeno, priradíme mu desiatkovú hodnotu(A-10 … F-15), ak je to číslo priradíme mu jeho hodnotu.
    3. Číslo v premennej **y** vynásobíme 16-kou a pripočítame k nemu hodnotu čísla z kroku 2. Postup opakujeme postupne pre všetky znaky 16-ového čísla.

7) 3.3c Previesť čísla medzi číselnými sústavami (3)

Zadanie:

- ❖ Vytvorte program na prevod čísla z inej sústavy do desiatkovej. Základ inej sústavy zadá používateľ a môže byť z intervalu 2 – 16.

```
z = 0
while z < 2 or z > 16:
    z = int(input("Zadaj základ sústavy 2 - 16: "))

s = input("Zadaj číslo v {} sústave: ".format(z))

dobre = True # ak je v zadanom reťazci chyba = false
y = 0 # výsledok - číslo v desiatkovej sústave

for c in s:

    #určenie desiatkovej hodnoty hexadecimalneho symbolu
    if c >= '0' and c <= '9':
        x = int(c)
    elif c >= 'a' and c <= 'f':
        x = ord(c) - ord('a') + 10
    elif c >= 'A' and c <= 'F':
        x = ord(c) - ord('A') + 10
    else:
        dobre = False # našiel sa nepovolený znak

    if x >= z:
        dobre = False; # našla sa číslica, ktorá do zadanej sústavy nepatrí

    if dobre:
        y = y * z + x
    else:
        break

if dobre:
    print("Číslo v desiatkovej sústave je ", y)
else:
    print("Zadané číslo obsahuje nepovolený znak, ktorý sa v {}ovej sústave
nepoužíva".format(z))
```

Tip na riešenie:

- ☞ Pre inú ako 16-ovú sústavu bude algoritmus rovnaký ako v úlohe 6.
- ☞ Pozri Tip na riešenie úlohy č. 6. Zmena bude len v tom, že pri výpočte násobíme medzivýsledok **y** základom sústavy **z:** $y = y * z + x$
- ☞ Do riešenia pridajte kontrolu, či každý znak v zadanom čísle patrí do danej sústavy.

8) *3.4a Získať vstup zo vstupného zariadenia, c vypísať textový a nakresliť grafický výstup*

Zadanie:

- Vytvorte program ktorý vykreslí grafické okno a na pozíciu ukazovateľa myši vykreslí po stlačení klávesu S -štvorec, K – Kruh, T – Vypíše text „Ahoj“.

```
import tkinter

canvas = tkinter.Canvas()
canvas.pack()

def kruh(suradnice):
    x = suradnice.x
    y = suradnice.y
    canvas.create_oval(x - 20, y - 20, x + 20, y + 20, fill="red")

def stvorec(suradnice):
    x = suradnice.x
    y = suradnice.y
    canvas.create_rectangle(x - 20, y - 20, x + 20, y + 20, fill="blue")

def text(suradnice):
    x = suradnice.x
    y = suradnice.y
    canvas.create_text(x, y, text="Ahoj!", fill="green")

canvas.bind_all("k", kruh)
canvas.bind_all("s", stvorec)
canvas.bind_all("t", text)

canvas.mainloop()
```

Tip na riešenie:

- Grafické okno sa vytvorí pomocou knižnice tkinter:

  `canvas = tkinter.Canvas()`

  `canvas.pack()`
- Obdĺžnik a elipsa sa kreslia príkazom:

  `canvas.create_rectangle(x1, y1, x2, y2, fill="farba")`

  `canvas.create_oval(x1, y1, x2, y2, fill="farba")`

  Objekt sa vykreslí do plochy obdĺžnikového tvaru z ľavým horným bodom $[x1, y1]$ a pravým dolným bodom $[x2, y2]$.
- Reakciu na stlačenie klávesy dosiahneme priradením udalosti funkcii, ktorá sa pri jej stlačení vykoná:

  `canvas.bind_all("klávesa", funkcia_ktorá_sa_po_stlačení_zavolá)`

9) 3.4b Spracovať vstup, ak je počet vstupných hodnôt dopredu neznámy (1)

Zadanie:

- Vytvorte program, ktorý vypočíta aritmetický priemer zadaných známok. Zadávanie známok sa ukončí zadaním čísla 0.

```
sucet = 0
pocet = 0

znamka = int(input("Zadaj {}. známku: ".format(pocet + 1)))
while znamka != 0:

    pocet = pocet + 1
    sucet = sucet + znamka
    znamka = int(input("Zadaj {}. známku: ".format(pocet + 1)))

if pocet > 0:
    print("Aritmetický priemer známok je ", sucet / pocet)
else:
    print("Nezadal si žiadnu známku!")
```

Tip na riešenie:

- Aritmetický priemer určite poznáte. Potrebujete **počet** známok a ich **súčet**.
  $Priemer = súčet / počet$
- Textové reťazce môžu na svoje formátovanie používať metódu `format` (metóda je jednoducho povedané funkcia, ktorá pracuje s objektom. Tu je objektom reťazec. Čo je to objekt a metóda sa naučíte neskôr, teraz si stačí zapamätať, ako sa použije). Funguje to tak, že každá dvojica zložených zátvoriek {} v reťazci sa postupne nahradí parametrami metódy `format` a takýto upravený reťazec vráti. `Format` dokáže oveľa viac, pozrite sa do manuálu k Pythonu.
- Metóda `format` sa používa takto: `"reťazec".format(parametre)`
  Napríklad: `"Fero má {} rokov a {} sestry".format(16,2)` vytvorí reťazec „*Fero má 16 rokov a 2 sestry*“.

10) 3.4b Spracovať vstup, ak je počet vstupných hodnôt dopredu neznámy (2)

Zadanie:

- Vytvorte program, ktorý vypočíta vážený priemer zo zadaných percentuálnych hodnotení s váhami. Zadávanie sa ukončí zadaním čísel 0, 0.

```
sucet_vah = 0
sucet_percent = 0
pocitadlo = 1

while True:
    p = int(input("Zadaj pocet percent {}. hodnotenia :
".format(pocitadlo)))
    v = int(input("Zadaj vahu {}. hodnotenia : ".format(pocitadlo)))
    if p == 0 and v == 0:
        break

    sucet_vah += v
    sucet_percent += v * p
    pocitadlo += 1

if sucet_vah > 0:
    print("Vážený priemer hodnotení je {} % ".format(sucet_percent /
sucet_vah))
else:
    print("Nezadal si žiadne hodnotenie!")
```

Tip na riešenie:

- Vážený priemer sa líši od aritmetického priemeru tým, že každé číslo sa do váženého priemeru započíta s určenou váhou. Používa sa napríklad pri maturitnej skúške a používajú ho aj niektorí vaši vyučujúci na výpočet výsledného hodnotenia.

  Vážený priemer **jednotky** s váhou 2 a **dvojky** s váhou 4 vypočítame $p = \frac{\mathbf{1}.2+\mathbf{2}.4}{2+4}$.

  Všeobecne pre 3 hodnotenia $\mathbf{h_i}$ s váhou $\mathbf{v_i}$ sa vážený priemer vypočíta takto:

$$p = \frac{h_1.v_1 + h_2.v_2 + h_3.v_3}{v_1 + v_2 + v_3}$$

  Pre **n** hodnotení:

$$p = \frac{h_1.v_1 + h_2.v_2 + \cdots + h_n.v_n}{v_1 + v_2 + \cdots + v_n}$$

11) 3.4b Spracovať vstup, ak je počet vstupných hodnôt dopredu neznámy (3)

Zadanie:

- Vytvorte program Kávomat. Do kávomatu „vhadzujete mince“ – zadávate hodnoty mincí 1; 2; 5; 10; 20; 50 Centov. Po obdržaní potrebnej sumy program „ponúkne kávu“ (vypíše text) a vypíše výdavok.

```
cena_kavy = 45
povolene_mince =(1, 2, 5, 10, 20, 50)

print("Dobrý deň, ponúknite sa kávou za {} centov, vhadzujte mince:
".format(cena_kavy))
print(povolene_mince)

suma = 0

while suma <= cena_kavy:
    minca = int(input("Treba zaplatiť {} centov. Vhoďte mincu:
".format(cena_kavy - suma)))

    if minca in povolene_mince:
        suma += minca
    else:
        print("Vhodili ste nesprávnu mincu, skúste znova!")

print ("Nech sa páčí, Vaša káva :) ")

vydavok = suma - cena_kavy

if vydavok > 0:
    print("Výdavok je {} centov".format(vydavok))
```

Tip na riešenie:

- Skontrolujte, či bola vhodená správna minca.
- Ak chcete zistiť, či premenná obsahuje hodnotu z určitého zoznamu hodnôt použite výraz `premenná in (hodnota_1, hodnota_2, …, hodnota_n)`. Výraz vracia True, ak sa hodnota premennej nachádza v zozname. Ináč vracia False.

12) 3.5a Zistiť, či je znak písmeno alebo cifra (1)

Zadanie:

- ❖ Napíšte program, ktorý v zadanom texte zistí, koľko je v ňom číslic, malých písmen a veľkých písmen.

```
text = input("Zadaj text: ")

cislice = 0
male = 0
velke = 0

for c in text:
    if c >= '0' and c <= '9':
        cislice += 1
    elif c >= 'A' and c <= 'Z':
        velke += 1
    elif  c >= 'a' and c <= 'z':
        male += 1

print("Číslic je ", cislice)
print("Malých písmen je", male)
print("Veľkých písmen je", velke)
```

Tip na riešenie:

- ☞ Operátory porovania <, <=, >,>= môžete použiť aj na porovnávanie reťazcov alebo znakov. Porovnávajú kódy (poradové číslo zanku) v tabuľke znakov. Platí: ‘A‘ < ‘Z‘ ,‘a‘ < ‘z‘ len pre písmená bez diakritiky. Písmená s diakritikou sa takto porovnávať nedajú. (Neplatí ‘a‘ < ‘á‘ < ‘č‘ < ‘d‘ …)

13) 3.5a Zistiť, či je znak písmeno alebo cifra (2)

Zadanie:

- Napíšte program, ktorý prečíta reťazec tvaru: číslicaOPERÁCIAčíslica a vypočíta ho. napr. 1+3 alebo 5 - 6. Ak zadá používateľ nesprávny vstup, program ho upozorní.

```
priklad = input("Zadaj príklad")

def cislica(c):
    if  c >= '0'and c <= '9':
        return ord(c) - ord('0')
    else:
        return -1

def operacia(c):
    if c in ('+','-','*','/'):
        return c
    else:
        return -1

dobre = True

x = cislica(priklad[0])
if x == -1:
    dobre = False

o = operacia(priklad[1])
if o == -1:
    dobre = False

y = cislica(priklad[2])
if y == -1:
    dobre = False

if dobre:
    if o == '+':
        z = x + y
    elif o == '-':
        z = x - y
    elif o == '*':
        z = x * y
    else:
        z = x / y

    print(x, o, y," = ", z)
else:
    print("V zadaní príkladu je chyba!")
```

Tip na riešenie:

- Znak z reťazca môžete vybrať `reťazec[index]`. Index pre prvý znak je 0.
- Číselnú hodnotu znaku predstavujúceho číslo napr. ‘5‘ získame odčítaním $ord(\text{‘5‘}) - ord(\text{‘0‘})$

14) 3.5b Zistiť výskyt znaku alebo podreťazca v textovom reťazci podľa daného kritéria: prvý, posledný, počet výskytov a pod. (1)

Zadanie:

- Napíšte program Veta, ktorý načíta zo vstupu vetu a skontroluje, či je jej prvé písmeno veľké a na konci je bodka.

```
veta = input("Zadaj vetu: ")
pismeno = veta[0]
bodka = veta[-1]
if pismeno >= "A" and pismeno <= "Z" and bodka == ".":
    print("Veta je v správnom formáte")
else:
    print("Veta nie je v správnom formáte")
```

Tip na riešenie:

- A chceme vybrať posledné písmeno z reťazca použijeme výraz `retazec[-1]`. Index -1 znamená „prvý znak od konca reťazca“.
- Upravte program tak, aby vracal správnu odpoveď aj pre písmená s diakritikou. V podmienenom príkaze if môžete použiť výraz `pismeno in retazec`. „Reťazec“ bude obsahovať všetky veľké písmená, aj tie s diakritikou.

15) 3.5b Zistiť výskyt znaku alebo podreťazca v textovom reťazci podľa daného kritéria: prvý, posledný, počet výskytov a pod. (2)

Zadanie:

- Napíšte program, ktorá spočíta počet predložiek „v“ a „na“ v zadanom texte. Po predložke musí nasledovať medzera.

```
text = input("Zadajte text: ")
if text[0:3] in ("Na ", "na ") :
    pocet_na = 1
else:
    pocet_na = 0

if text[0:2] in ("V ", "v ") :
    pocet_v = 1
else:
    pocet_v = 0

for index in range(len(text)):
    if text[index:index + 4] in (" na ", " Na "):
        pocet_na += 1
    if text[index:index + 3] in (" v ", " V "):
        pocet_v += 1

print("Počet predložiek \"na\" je", pocet_na)
print("Počet predložiek \"v\" je", pocet_v)
```

Tip na riešenie:

- Hoci Python obsahuje funkciu na spočítanie výskytu podreťazca v danom reťazci, pokúste sa vyriešiť úlohu bez nej.
- V našom riešení sme v cykle vybrali podreťazec dĺžky 4 od pozície index v reťazci `text[index:index + 4]` a zistili sme operátorom `in` , či sa tento podreťazec vyskytuje v zozname predložiek `(" na ", " Na ")`.
- Nezabudnite ošetriť prípad, že predložka je na začiatku reťazca.

16) 3.5c Nahradiť alebo odstrániť znak alebo podreťazec v textovom reťazci.

Zadanie:

- Vytvorte program Test zo slovenčiny, ktorý v texte nahradí všetky formy písmen **i** a **y** (**I**, **Í**, **i**, **í**, **Y**, **Ý**, **y**, **ý**) podtržítkom a takýto pozmenený text vypíše.

```
text=input("Zadajte text: ")
for pismeno in text:
    if pismeno in "IÍiíYÝyý" :
        print("_", end = "")
    else:
        print(pismeno, end ="")
```

Tip na riešenie:

- ☞ Na zistenie, či sa písmeno nachádza v reťazci použijeme výraz písmeno in retazec. Výraz vracia logickú hodnotu True / False.
- ☞ Príkaz print(„Ahoj“) vypíše text „Ahoj“ a presunie kurzor na nový riadok. Prechod na nový riadok sa dá potlačiť nastavením parametra `end` v príkaze `print` na ‘‘

17) 3.5d Zostaviť textový reťazec z podreťazcov podľa daných kritérií.

Zadanie:

- ❖ Vytvorte program Heslo, ktorý vygeneruje náhodné heslo pre používateľa podľa daných kritérií: Heslo má 8 znakov a striedajú sa v ňom náhodné hlásky: spoluhláska-samohláska-spoluhláska...

```
import random
heslo = ""
for i in range(4):
    heslo += random.choice("bcdfghklmnprstvwxz")
    heslo += random.choice(("a", "e", "i", "o", "u", "y"))

print("Heslo je: ", heslo)
```

Tip na riešenie:

- ☞ Náhodný znak môžeme vybrať z reťazca znaky príkazom `random.choice('znaky')`.
- ☞ Funkcia `choice` sa nachádza v module `random`, ktorý treba pred použitím sprístupniť príkazom `import random`.
- ☞ Pozn. V našom riešení prvý `choice` vyberá 1 znak z reťazca, druhý `choice` vyberá 1 položku zo zoznamu – príklad dvoch rôznych spôsobou použitia choice.

18) 3.5 d Zostaviť textový reťazec z podreťazcov podľa daných kritérií.

Zadanie:

- ❖ Vytvorte program Heslo, ktorý vygeneruje heslo pre používateľa podľa daných kritérií: Heslo má 10 znakov a strieda sa v ňom číslica-písmeno-číslica...

```
import random
heslo = ""
for i in range(5):
    heslo += str(random.randint(0, 9))
    heslo += chr(random.randint(ord("a"),ord("z")))
print("Heslo je: ", heslo)
```

Tip na riešenie:

- ☞ Pozri Tip na riešenie k úlohe 17.
- ☞ V ukážke sme použili funkciu `randint(a, b)`, z modulu random, ktorá generuje celé číslo v intervale od `a` po `b` vrátane.
- ☞ Ak chceme číslo pridať vygenerované číslo do reťazca, je najprv potrebná konverzia čísla reťazec funkciou `str(cislo)`.
- ☞ Náhodne vygenerované poradové číslo znaku z tabuľky znakov (tzv. ordinárne číslo znaku) konvertujeme na reťazec funkciou `chr(cele_cislo)`.

19) 3.5e Formátovať výstup (1).

Zadanie:

- ❖ Napíšte program Vizitka, ktorý načíta meno človeka a vykreslí zo znakov vizitku v tvare:

  ****

  * Ján Botto *

  ****

```
meno=input("Zadaj meno: ")
dlzka = len(meno)
print('*' * (dlzka + 4))
print('* ' + meno + " *")
print('*' * (dlzka + 4))
```

Tip na riešenie:

- ☞ V Pythone sa dajú reťazce nielen sčítavať – zreťazovať, ale aj násobiť celým číslom. Vynásobením reťazca číslom 3 `'Ahoj ' * 3` získame reťazec ‘Ahoj Ahoj Ahoj‘.
- ☞ Pokúste sa naprogramovať túto úlohu bez použitia násobenia reťazca číslom. Použite napr. cyklus for.

20) 3.5e Formátovať výstup (2).

Zadanie:

- Napíšte program Krajšia vizitka, ktorý načíta meno a adresu a zobrazí vizitku v tvare:

```
+-------------------------+
| Janko Hraško            |
| Malinová 12, Ružomberok |
+-------------------------+
```

```
meno = input("Zadaj meno: ")
adresa = input("Zadaj adresu: ")

if len(meno) > len(adresa):
    dlzka = len(meno)
else:
    dlzka = len(adresa)

print("+-" + "-" * dlzka + "-+")
print("| " + meno + " " * (dlzka - len(meno)) + " |")
print("| " + adresa +  " " * (dlzka - len(adresa)) + " |")
print("+-" + "-" * dlzka + "-+")
```

Tip na riešenie:

- Pozri Tip Na riešenie k úlohe 19.
- Nezabudnite ošetriť prípad, že meno môže byť dlhšie ako adresa alebo opačne.

21) 3.6a Vymeniť hodnoty dvoch premenných.

Zadanie:

- Program načíta mená a vek dvoch ľudí (prvy_meno, prvy_vek, druhy_meno, druhy_vek) a vypíše vetu v tvare „Janko **nie je mladší ako** Jurko“. Obmedzenie na riešenie úlohy: príkaz na výpis môže byť použitý v programe len na jednom mieste a musí byť:
  `print(prvy_meno, " nie je mladsi ako ", druhy_meno)`

```
prvy_meno = input("Zadaj meno prvého človeka: ")
prvy_vek  = int(input("Zadaj vek prvého človeka:"))
druhy_meno = input("Zadaj meno druhého človeka: ")
druhy_vek  = int(input("Zadaj vek druhého človeka:"))

if druhy_vek > prvy_vek:
    pomocna = prvy_meno
    prvy_meno = druhy_meno
    druhy_meno = pomocna

print(prvy_meno, " nie je mladsi ako ", druhy_meno)
```

Tip na riešenie:

- ☞ Úlohu riešte pmocou výmeny hodnoty dvoch premenných. V Pythone sú dve možnosti:
  - pomocou pomocnej premennej: `c = a, a = b, b = c`,
  - priame priradenie: `a, b = b, a`.

22) 3.7a zostaviť podmienky pre vetvenie a cyklus podľa zadania úlohy, prechádzať prvky postupnosti (1).

Zadanie:

- ❖ Vytvorte program, ktorý vypíše všetky delitele zadaného prirodzeného čísla.

```
cislo = int(input("Zadaj cislo: "))
print("Zoznam deliteľov:")
for delitel in range(1, cislo + 1):
    if cislo % delitel == 0:
        print(delitel)
```

Tip na riešenie:

- ☞ Deliteľ čísla **x** je číslo **d**, ktoré delí číslo **x** bezo zvyšku (zvyšok je nula) `x % d == 0`

23) 3.7a Zostaviť podmienky pre vetvenie a cyklus podľa zadania úlohy, prechádzať prvky postupnosti (2).

Zadanie:

- ❖ Vytvorte program, ktorý zistí a vypíše, či je zadané celé číslo prvočíslo.

```
cislo = int(input("Zadaj cislo "))
je_prvocislo = cislo > 1
for delitel in range(2, cislo):
    if cislo % delitel == 0:
        je_prvocislo = False
        break

if je_prvocislo:
    print("Číslo", cislo, "je prvočíslo.")
else:
    print("Číslo", cislo, "nie je prvočíslo.")
```

Tip na riešenie:

- ☞ Prvočíslo je číslo, ktoré nemá iné delitele okrem jednotky a samého seba.
  O deliteľnosti pozri Tip na riešenie k úlohe 22.
- ☞ Nami uvedený algoritmus nie je veľmi efektívny, pretože zbytočne testuje delitele väčšie ako odmocnina z testovaného čísla. Hornú hranicu cyklu môžeme menšiť na **odmocninu z testovaného čísla + 1**: `int(math.sqrt(cislo)) + 1`
- ☞ Funkcia `sqrt(x)` je druhá odmocnina z `x` a nachádza sa v module **math**, ktorý treba naimportovať.

24) 3.7c Akumulovať výsledky podľa daných kritérií.

Zadanie:

- ❖ Vytvorte program Bankomat. Bankomat vydáva bankovky 10 €, 20 €, 50 €, 100 €. Načíta sumu, ktorú má vyplatiť a vypíše druh a počet bankoviek ktoré vydá. Ak sa suma nedá vyplatiť (napr. 25 €) oznámi to.

```
pocet_100 = 0
pocet_50  = 0
pocet_20  = 0
pocet_10  = 0

suma = int(input("Zadajte sumu, ktorú vám vyplatím: "))

while( suma >= 100):
    suma -= 100
    pocet_100 += 1

while( suma >= 50):
    suma -= 50
    pocet_50 += 1

while( suma >= 20):
    suma -= 20
    pocet_20 += 1

while( suma >= 10):
    suma -= 10
    pocet_10 += 1

if suma == 0:
    print("Počet 100 € - ", pocet_100)
    print("Počet  50 € - ", pocet_50)
    print("Počet  20 € - ", pocet_20)
    print("Počet  10 € - ", pocet_10)
else:
    print("Prepáčte, obnos sa nedá vyplatiť.")
```

Tip na riešenie:

- ☞ Obnos vyplácajte v bankovkách čo najväčšej hodnoty.

25) 3.7d Vyberať hodnoty, ktoré spĺňajú dané kritériá.

Zadanie:

- ❖ Vytvorte program Terč, ktorý do grafického okna nakreslí 200 farebných krúžkov na náhodnej pozícii. Ak je krúžok vzdialený od stanoveného bodu menej ako 50 (terč) bude červený, ak ďalej ako 50 bude modrý.

```
import tkinter
import random

xmax = 200
ymax = 200
stred_x = 100
stred_y = 100

canvas = tkinter.Canvas(height=ymax, width=xmax)
canvas.pack()


for i in range(200):
    x = random.randint(1,xmax)
    y = random.randint(1,ymax)

    if (x - stred_x)**2 + (y - stred_y)**2 < 50**2:
        canvas.create_oval(x - 5, y - 5, x + 5, y + 5, fill="red")
    else:
        canvas.create_oval(x - 5, y - 5, x + 5, y + 5, fill="blue")
canvas.mainloop()
```

Tip na riešenie:

- ☞ Bod $[x, y]$ sa nachádza vo vzdialenosti menej ako 50 od bodu $[Sx, Sy]$, ak platí vzťah: $(x - S_x)^2 + (y - S_y)^2 < 50^2$
- ☞ Náhodné celé číslo z intervalu $< 1, max >$ vygenerujeme príkazom `x = random.randint(1, max)`, predtým je potrebné importovať do programu knižnicu `random`.
- ☞ Pozri tiež Tip k úlohe 8, práca s grafikou.

26) 3.7e Používať vnorené programové konštrukcie.

Zadanie:

- Vypíšte na obrazovku tabuľku malej násobilky. V záhlaví riadkov aj stĺpcov budú čísla 1 až 10. V bunkách tabuľky bude súčin týchto čísel.

```
for riadok in range(1, 11):
    for stlpec in range(1, 11):
        sucin = stlpec * riadok
        print("{:4}".format(sucin), end="")
    print()
```

Tip na riešenie:

- Aby bol výstup pekne sformátovaný použijeme vo funkcii print metódu format, ktorá vytvára formátovaný reťazec.
  - `"{:4}".format(sucin)`. Formátovací reťazec `{:4}` znamená, že na miesto zátvoriek sa doplní parameter metódy `format` a doplní sa medzerami na 4 znaky.
- V príkaze `print` priradenie prázdneho reťazca parametru `end` spôsobí, že po výpise sa neprejde na nový riadok. (Parameteru `end` môžeme priradiť ľubovoľný reťazec, ktorým sa výpis ukončí).

27) 3.7f Rozhodovať sa kedy stačí použiť cyklus s pevným počtom opakovaní, a kedy treba cyklus s podmienkou.

Zadanie:

- Vytvorte program, ktorý z klávesnice načíta hodnotenie štyroch žiakov a vypočíta ich priemer známok. Každý žiak môže mať ľubovoľný počet známok, za poslednou známkou žiaka je zadaná nula.

  Príklad. vstup: 1 1 1 1 0 2 3 0 2 0 3 3 0 výstup: 1.0, 2.5, 2.0, 3.0

```
for i in range(4):
    sucet = 0
    pocet = 0
    znamka = int(input("Zadaj známku {}. žiaka: ".format(i+1)))
    while znamka > 0:
        sucet = sucet + znamka
        pocet = pocet + 1
        znamka = int(input("Zadaj známku {}. žiaka: ".format(i + 1)))
    if pocet > 0:
        print("Priemer {}. žiaka je {}".format(i+1, sucet/pocet))
    else:
        print("Nezadal si známku!")
```

Tip na riešenie:

- V riešení sú dva cykly, jeden je vnorený v druhom. Ktorý je s dopredu známym počtom opakovaní a ktorý s podmienkou?

28) 3.8a Generovať čísla v danom rozsahu.

Zadanie:

- ❖ Vytvorte hru Hádaj číslo. Počítač náhodne zvolí celé číslo z intervalu <0,1000>. Hráč ho postupne háda. Po každom hráčovom type počítač vypíše, či si počítač zvolil väčšie alebo menšie číslo. Po uhádnutí čísla hra končí.

```
import random
pokus = 0
hadaj = random.randint(1, 1000)
print("Myslím si celé číslo od 1 do 1000. Ty ho budeš hádať")
pokracuj = True
while pokracuj:
    pokus += 1
    cislo = int(input("Zadaj tvoj tip: "))

    if hadaj > cislo:
        print("Myslím si väčšie číslo.")
    elif hadaj < cislo:
        print("Myslím si menšie číslo.")
    else:
        print("Uhádol si na", pokus," pokus")
        pokracuj = False
```

Tip na riešenie:

- ☞ V našom riešení sme použili booleovskú premennú `pokracuj`, ktorá sa testuje na začiatku cyklu while. Ak je `True` cyklus , v ktorom sa načíta hráčov tip, pokračuje. Ak hráč uhádne číslo, nastaví sa na `False` a cyklus sa ukončí.
- ☞ Pri testovaní hráčovho typu sme použili podmienený príkaz `if-elif-else`
- ☞ Ak podmienka v `if` neplatí testuje sa podmienka v `elif` a ak ani tá neplatí vykonajú sa príkazy za `else`.

29) 3.8b Simulovať danú činnosť (napríklad: zostavovať frekvenčnú tabuľku pri hode dvoma kockami a pod.).

Zadanie:

- Vytvorte program, ktorý bude simulovať hádzanie dvoma kockami a zaznamenávať súčet padnutých čísel-bodov. Na konci zobrazí stĺpcový diagram počtu padnutých bodov. (Pozn. diagram bude obsahovať stĺpčeky pre 2 až 12 bodov. Výška stĺpčeka bude úmerná počtu, koľkokrát padol daný počet bodov)

```
import random
import tkinter

tabulka = [0] * 12
max = 0

canvas = tkinter.Canvas()
canvas.pack()

def stlpec(cislo, pocet, max):
    vyska = pocet / max * 100
    canvas.create_rectangle(cislo * 20, 120, cislo *20 + 18, 120 -vyska,
fill="blue")
    canvas.create_text(cislo * 20 + 10 , 130,text=str(cislo))

pocet_hodov = 20

for i in range(pocet_hodov):
    k1 = random.randint(1, 6)
    k2 = random.randint(1, 6)

    k = k1 + k2
    tabulka[k - 1] += 1
    if tabulka[k - 1] > max:
        max = tabulka[k - 1]

for i in range(1,13):
    stlpec(i, tabulka[i - 1], max)

canvas.mainloop()
```

Tip na riešenie:

- Koľkokrát padne určitý počet bodov budeme uchovávať v jednorozmernom poli tabulka. V Pythone sa nazýva **zoznam**.
- Na dvoch kockách môže padnúť max. 12 bodov. Teda počet položiek poľa zvolíme 12.
- Na vykreslenie stĺpca diagramu vytvoríme funkciu stlpec, ktorej odovzdáme parametre: cislo – počet bodov, ktoré padli, pocet – koľkokrát padol daný počet bodov a max - najviac koľko krát padol určitý počet bodov.

30) 3.8c Generovať náhodnú kresbu podľa stanovených kritérií (napríklad: náhodne rozmiestnené geometrické tvary, kreslenie obdĺžnika z veľkého počtu náhodne zafarbených úsečiek a pod.) (1).

Zadanie:

- ❖ Vytvorte program ktorý vykreslí 20 náhodne rozmiestnených štvorcov náhodnej farby, prípadne aj náhodnej veľkosti.

```
import tkinter
import random

xmax = 300
ymax = 300

canvas = tkinter.Canvas(width=xmax, height=ymax)
canvas.pack()


for i in range(20):
    strana = random.randint(10,100)
    x = random.randint(0, xmax)
    y = random.randint(0, ymax)
    farba = random.choice(("black", "red", "blue", "yellow", "green"))
    canvas.create_rectangle(x, y, x + strana, y + strana, fill=farba)

canvas.mainloop()
```

Tip na riešenie:

- ☞ Náhodné generovanie rozmerov môžete použiť funkciu `randint(min, max)`, ktorá vygeneruje celé náhodné číslo od `min` po `max`.
- ☞ Náhodnú farbu môžeme vybrať funkciou `choice(položka1, položka2, …)`, ktorá náhodne vyberie jednu z položiek `položka1`, `položka2` … .
- ☞ Funkcia choice a randint sa nachádzajú v module `random`, preto ho treba na začiatku programu importovať.

31) 3.8 c generovať náhodnú kresbu podľa stanovených kritérií (napríklad: náhodne rozmiestnené geometrické tvary, kreslenie obdĺžnika z veľkého počtu náhodne zafarbených úsečiek a pod.) (2)

Zadanie:

- Nakreslite „trávnik“ ktorý bude vytvorený zo zvislých úsečiek náhodnej dĺžky maximálne 50. Steblá trávy budú pozdĺž dolného okraja grafického okna. Farba stebla bude zelená alebo žltá, v závislosti od dĺžky stebla. Kratšie ako 20 budú žlté, dlhšie zelené.

```
import tkinter
import random

canvas = tkinter.Canvas()
canvas.pack()


for i in range(0,300,2):
    dlzka = random.randint(1,50)
    if dlzka < 20:
        farba = "yellow"
    else:
        farba = "green"

    canvas.create_line(i,95,i,95 - dlzka,fill=farba,width=2)

canvas.mainloop()
```

Tip na riešenie:

- ☞ Skúste riešenie doplniť o nakreslenie kvetinky, ak steblo bude dlhšie ako 40.

32) 3.9a Definovať vlastné podprogramy s parametrami. (1)

Zadanie:

- Vyriešte úlohu 30 definovaním vlastnej funkcie „stvorec(x, y, a, farba)“, ktorý nakreslí štvorec na pozícii „x,y“ (ľavý horný roh) so stranou „a“ a s farbou „farba“.

```
import tkinter
import random

xmax = 300
ymax = 300

canvas = tkinter.Canvas(width=xmax, height=ymax)
canvas.pack()

def stvorec(x, y, a, farba):
    canvas.create_rectangle(x, y, x + a, y + a, fill=farba)

for i in range(20):
    strana = random.randint(10,100)
    x = random.randint(0, xmax)
    y = random.randint(0, ymax)
    farba = random.choice(("black", "red", "blue", "yellow", "green"))
    stvorec(x, y, strana, farba)

canvas.mainloop()
```

Tip na riešenie:

- ☞ Upravte program tak, aby funkcia stvorec prijímala len parametre x, y a na tejto pozícii vykreslila štvorec náhodnej farby a veľkosti.

33) 3.9a Definovať vlastné podprogramy s parametrami. (2)

Zadanie:

- Vytvorte program, ktorý nakreslí 5 rôznych eurocentových mincí (kruh s číslom hodnoty mince) na súradniciach x, y pomocou funkcie `minca(x , y, hodnota)`. Priemer mince nech závisí od hodnoty mince.

```
import tkinter

canvas =  tkinter.Canvas()
canvas.pack()

def minca(x, y, hodnota):
    polomer = 10 + hodnota // 2
    canvas.create_oval(x - polomer, y - polomer, x + polomer, y + polomer,
fill = "orange", width = 3)
    canvas.create_text(x, y, text=hodnota)

x = 20
for i in 2,5,10,20,50:
    minca(x , 50, i)
    x = x + 30 + 2 * i

canvas.mainloop()
```

Tip na riešenie:

- ☞ Porozmýšľajte, ako vypočítate polomer mince z jej hodnoty.
- ☞ Zvážte tiež, ako sa má meniť x-ová súradnica mince v závislosti od polomeru, aby sa mince neprekrývali.

34) 3.9b Definovať vlastné funkcie s návratovou hodnotou rôznych typov (čísla, texty, logické hodnoty a pod.). (1)

Zadanie:

- ❖ Vytvorte program na výpočet najväčšieho deliteľa dvoch zadaných celých čísel. Vytvorte a použite v ňom funkciou `delitel`, ktorá vráti najväčší spoločný deliteľ čísel zadaných z klávesnice.

```
def delitel(a, b):
    while a != b:
        if a > b:
            a = a - b
        else:
            b = b - a
    return a

a = int(input("Zadaj prvé číslo: "))
b = int(input("Zadaj druhé číslo: "))
d = delitel(a, b)
print("Najväčší spoločný deliteľ čísel {} a {} = {}".format(a, b, d))
```

Tip na riešenie:

- ☞ Na hľadanie najväčšieho spoločného deliteľa existuje jednoduchá verzia Euklidovho algoritmu:
    1. Máme dve celé kladné čísla a a b, ktorých n.s.d. chceme vypočítať.
    2. Ak **a** je väčšie ako **b**, tak od **a** odčítame **b** a uložíme do **a**.
    3. Ak je **b** väčšie ako **a**, tak od **b** odčítame **a** a uložíme do **b**.
    4. Kroky 2. - 4. opakujeme, až kým sa čísla **a** a **b** nerovnajú.
    5. Najväčší spoločný deliteľ je v **a** (samozrejme aj v **b**) .
- ☞ Návratová hodnota `a` funkcie sa priraďuje príkazom `return a`, tento príkaz zároveň spôsobí ukončenie funkcie a návrat na miesto v programe, z ktorého bola funkcia vyvolaná.

35) 3.9b Definovať vlastné funkcie s návratovou hodnotou rôznych typov (čísla, texty, logické hodnoty a pod.). (2)

Zadanie:

- Vytvorte program na zistenie, či je dané číslo prvočíslo. Vytvorte a použite v ňom funkciu `je_prvocislo(x)`, s jedným parametrom **x**. Funkcia vráti logickú hodnotu **True**, ak je číslo **x** prvočíslo, alebo vráti **False**, ak **x** nie je prvočíslo.

```
def je_prvocislo(x):
    je = x > 1
    for delitel in range (2,x):
        if x % delitel == 0:
            je = False
            break
    return je

cislo = int(input("Zadaj číslo: "))
if je_prvocislo(cislo):
    print("Číslo je prvočíslo")
else:
    print("Číslo nie je prvočíslo")
```

Tip na riešenie:

- Upravte funkciu `je_prvocislo`, ktorá vráti 0, ak je číslo prvočíslo, alebo vráti počet deliteľov (okrem jednotky a samého seba), ak prvočíslo nie je. Výsledok funkcie s príslušným komentárom vypíšte.
- Pozri tiež tipy k úlohám 22 a 23.

36) 3.9b Definovať vlastné funkcie s návratovou hodnotou rôznych typov (čísla, texty, logické hodnoty a pod.) (3).

Zadanie:

- Vytvorte program na výpočet koreňov kvadratickej rovnice pomocou diskriminantu. Na výpočet diskriminantu vytvorte funkciu `diskriminant(a, b, c)`, ktorý vráti hodnotu diskriminantu na základe parametrov **a**, **b**, **c**.

```
import math

def diskriminant(a, b, c):
    d = b * b - 4 * a * c
    return d

print("Zadaj koeficienty rovnice ax2 + bx + c = 0")
a = float(input("Zadaj a: "))
b = float(input("Zadaj b: "))
c = float(input("Zadaj c: "))

d = diskriminant(a, b, c)

if d > 0:
    x1 = (-b + math.sqrt(d)) / (2 * a)
    x2 = (-b - math.sqrt(d)) / (2 * a)
    print("Korene sú: x1 = ", x1,", x2 = ", x2)
elif d == 0:
    x = -b / (2 * a)
    print("Koreň je: x = ", x)
else:
    print("Rovnica nemá reálne korene")
```

Tip na riešenie:

- Pre tých, ktorí sa nekamarátia s matematikou:
  - Kvadratická rovnica má tvar $ax^2 + bx + c = 0$, x je neznáma a a, b, c sú čísla.
  - Rovnica sa rieši výpočtom diskriminantu: $D = b^2 - 4ac$.
  - Ak je diskriminant kladný rovnica má dve riešenia x_1 a x_2:

    $$x_1 = \frac{-b + \sqrt{D}}{2a}, x_2 = \frac{-b - \sqrt{D}}{2a}.$$

  - Ak je diskriminant rovný nule, existuje jedno riešenie $x = -\frac{b}{2a}$.
  - Ak je diskriminant záporný, rovnica nemá (reálne) riešenie.
  - Odmocnina z čísla x sa v Pythone vypočíta funkciou `math.sqrt(x)`, ktorá sa nachádza v module `math`. Modul `math` treba importovať do programu.

37) 3.10a generovať obsah poľa podľa daných kritérií (napríklad: vynulovanie poľa, každý nasledujúci prvok je súčtom predchádzajúcich dvoch a pod.). (1)

Zadanie:

- Zostavte program, ktorý vytvorí 20 prvkové pole obsahujúce Fibonacciho postupnosť: prvý prvok = 1, druhý prvok = 1, každý ďalší = súčet predchádzajúcich dvoch prvkov. Nakoniec pole vypíšte.

```
#vytvorenie poľa
pole = [0] * 20
pole[0] = pole[1] = 1

#naplnenie poľa hodnotami
for i in range(2,20):
    pole[i] = pole[i - 1] + pole[i - 2]

#výpis poľa
for i in range(20):
    print(pole[i])
```

Tip na riešenie:

- Pole je v Pythone dátová štruktúra nazývaná **zoznam**.
- Zoznam obsahujúci 20 položiek, ktoré sú inicializované nulou, sa vytvorí `pole = [0] * 20`.
- K položkám poľa sa pristupuje pomocou indexu `pole[index]`. Index nadobúda hodnoty od 0 po 19.

38) 3.10a generovať obsah poľa podľa daných kritérií (napríklad: vynulovanie poľa, každý nasledujúci prvok je súčtom predchádzajúcich dvoch a pod.). (2)

Zadanie:

- ❖ Zostavte program, ktorý vytvorí 10 prvkové pole. Prvkami poľa budú postupne celé kladné čísla, ktoré dávajú po delení troma zvyšok 2. Pole vypíšte.

```
#vytvorenie poľa
pole = [0] * 10
cislo = 2
#naplnenie poľa hodnotami
for i in range(10):
    pole[i] = cislo
    cislo += 3
#výpis poľa
for i in range(10):
    print(pole[i])
```

Tip na riešenie:

- ☞ Pozri tiež Tip k úlohe 37.
- ☞ Najmenšie kladné číslo, ktoré dáva po delení trojkou číslo 2 je dvojka.
- ☞ Každé ďalšie takéto číslo je o 3 väčšie.

39) 3.10b Používať pole na uchovanie väčšieho počtu údajov (napríklad: frekvenčná tabuľka výskytu nejakých hodnôt, súradnice bodov v rovine a pod.).

Zadanie:

- Vytvorte program na štatistické spracovanie výsledkov písomky. Na vstup zadávame postupne známky 1 – 5. Zadávanie ukončíme zadaním 0. Program vypíše počet jednotiek, dvojok, trojok, štvoriek a pätiek.

```
#vytvorenie poľa, v ktorom budú počty jednotlivých známok
#jednotky budy v pocty[0], dvojky v pocty[1] ...
pocty = [0] * 5

while True:
    znamka = int(input("Zadaj známku: "))
    if znamka == 0:
        break
    pocty[znamka - 1] += 1

for i in range(5):
    print("Známka", i + 1, " - počet:", pocty[i])
```

Tip na riešenie:

- Pozri tiež Tip k úlohe 37.
- Index položky v poli je číslovaný od 0. Aby sme pole využili, počet jednotiek bude v `pocty[0]`, dvojok v `pocty[1]` …

40) 3.10c Hľadať prvky poľa s danými vlastnosťami (napríklad: najmenší, druhý najmenší a pod.).

Zadanie:

- Vytvorte program na štatistické spracovanie výsledkov písomky: Načíta do poľa počty bodov za písomky a vypíše informáciu, o počte bodov najlepšej a najhoršej písomky. Načítanie známok sa ukončí zadaním čísla -1.

```
body = [] #vytvorenie prázdneho poľa bodov

# načítanie hodnôt poľa
while True:
    hodnotenie = int(input("Zadaj body z písomky : "))
    if hodnotenie == -1:
        break
    body.append(hodnotenie) #vložíme do poľa ďalšie hodnotenie

#spracovanie hodnôt poľa
min = max = body[0]
for b in body:
    if min > b:
        min = b
    if max < b:
        max = b

print("Najlepšia písomka má", max, "bodov.")
print("Najhoršia písomka má", min, "bodov.")
```

Tip na riešenie:

- Pozri tiež Tip k úlohe 37.
- Prázdny zoznam (bez položiek) sa vytvorí `pole=[]`
- Položka s hodnotou **hodnota** sa pridáva na koniec zoznamu príkazom `pole.append(hodnota)`

41) 3.10d Zisťovať, či pole obsahuje hodnoty s danými vlastnosťami (napríklad či obsahuje číslo 0, prvý výskyt medzery a pod.), resp. ich počet.

Zadanie:

- Zostavte program ktorý načíta zo vstupu text a zistí, koľko samohlások sa v ňom vyskytuje. Vstupný text obsahuje len slová bez diakritiky (bez mäkčeňov a dĺžňov).

```
text = input("Zadaj text: ")

pocet = 0

for znak in text:
    if znak in "aeiouyAEIUOY":
        pocet += 1

print("Počet samohlások v texte je", pocet)
```

Tip na riešenie:

- ☞ Rozšírte program aj o počítanie samohlások a spoluhlások s diakritikou

42) 3.10 e zisťovať, či pole spĺňa dané kritérium (napríklad: či sú všetky prvky rovnaké, rôzne, či sú hodnoty usporiadané a pod.).

Zadanie:

- Načítajte a uložte do poľa 20 čísel z klávesnice. Prejdite v cykle pole a prvky, ktoré sa vyskytujú práve raz, vypíšte.

```
POCET = 20
pole = [0] * POCET

#načítanie 20 čísel
for i in range(POCET):
    cislo = int(input("Zadaj číslo: "))
    pole[i] = cislo

print("Jedenkrát sa vyskytli čísla:")
for i in pole:
    pocet = 0
    for j in pole:
        if i == j :
            pocet = pocet + 1

    if pocet == 1:
        print(i)
```

Tip na riešenie:

- Jednoduchý algoritmus ktorý zisťuje, či sa prvok nachádza v poli práve n-krát obsahuje dva vnorené cykly. Obidva cykly postupne prechádzajú všetky prvky poľa. Pred začiatkom vnútorného cyklu nastavíme premennú `pocet` na 0. V tele vnútorného cyklu testujeme, či sa prvky riadiacich premenných obidvoch cyklov rovnajú. Ak áno zvýšime hodnotu premennej `pocet` o 1. Po ukončení vnútorného cyklu skontrolujeme, či sa hodnota premennej `pocet` rovná n.

43) 3.10f Modifikovať prvky poľa (napríklad: vsunúť prvok na dané miesto tak, aby ostali prvsky usporiadané – pozor, už nie triediace algoritmy)

Zadanie:

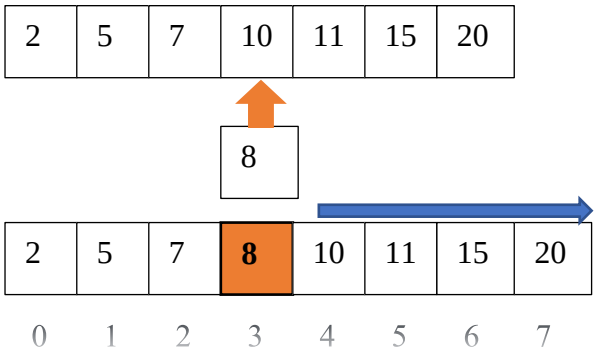
- Napíšte program ktorý bude postupne načítavať zo vstupu 20 čísel predstavujúcich výšku žiakov triedy 2.A v cm. Každú načítanú hodnotu uloží do poľa na správne miesto, tak aby pole bolo stále zoradené podľa rastúcej výšky žiaka.

```
POCET = 10

pole = [0] * POCET

for i in range(POCET):
    vyska = int(input("Zadaj výšku žiaka: "))

    j = 0
    while vyska > pole[j] and j < i: # najdenie prvku v poli, kde sa vlozi vyska
        j += 1

    while j <= i: #posun ostatnych prvkov doprava
        pom = pole[j]
        pole[j] = vyska
        vyska = pom
        j += 1

for x in pole:
    print(x)
```

Tip na riešenie:

- ☞ Po načítaní čísla, treba zistiť, za ktorý prvok poľa sa má toto číslo zaradiť.

  Prvky vpravo od neho sa posunú o jedno miesto vyššie.



44) *3.10g Manipulovať s viacerými poľami podľa daných kritérií (napríklad: kopírovanie časti poľa, otočenie, posunutie, zlučovanie dvoch usporiadaných postupností a pod.).*

Zadanie:

- Máme zadané dve polia **a** a **b,** ktoré obsahujú čísla usporiadané podľa veľkosti od najmenšieho po najväčšie. Zostavte program, ktorý vytvorí pole **c**, ktoré bude obsahovať prvky zlúčených polí **a** a **b** tak, aby bolo zoradené od najmenšieho po najväčší prvok.

```
a = (1, 5, 11, 15, 20, 24, 32)
b = (2, 3, 4, 6, 14, 25, 26, 31)

print("Pole a =", a)
print("Pole b =", b)

c = [] #výsledné pole

i = 0
j = 0

# zlúčenie polí do poľa c
while i < len(a) and j < len(b):
    if a[i] < b[j]:
        c.append(a[i])
        i += 1
    else:
        c.append(b[j])
        j += 1

# vloženie zbytku poľa (prebehne buď jeden alebo druhý cyklus)
while i < len(a):
    c.append(a[i])
    i += 1

while j < len(b):
    c.append(b[j])
    j += 1

print("Výsledné pole c =", c)
```

Tip na riešenie:

- ☞ Prechádzajte súčasne polia **a** a **b** a do poľa **c** vložte ten prvok, ktorý je na aktuálnej pozícii v poliach **a** a **b** menší.
- ☞ Porozmýšľajte, prečo v programe musí byť ešte časť kódu za **#vloženie zbytku poľa**…

45) 3.10g Manipulovať s viacerými poľami podľa daných kritérií (napríklad: kopírovanie časti poľa, otočenie, posunutie, zlučovanie dvoch usporiadaných postupností a pod.).

Zadanie:

- Zostavte program Slovník, ktorý vytvorí pole `slovnik` obsahujúci slová usporiadané podľa abecedy z polí `slovnik1` a `slovnik2` .Vstupné polia `slovnik1` a `slovnik2` sa načítajú z textových súborov **slovnik1.txt** a **slovnik2.txt** a sú už usporiadané podľa abecedy. Výsledné pole vypíšte a uložte do súboru **slovnik.txt**.

```
def nacitaj_pole(nazov_suboru):
    pole = []
    subor = open(nazov_suboru, 'r')
    riadok = subor.readline()
    while riadok != '':
        pole.append(riadok.strip())
        riadok = subor.readline()
    subor.close()
    return pole
def zluc_polia(a, b):
    i = 0
    j = 0
    pole = []
    while i < len(a) and j < len(b):
        if a[i] < b[j]:
            pole.append(a[i])
            i += 1
        else:
            pole.append(b[j])
            j += 1

    while i < len(a):
        pole.append(a[i])
        i += 1

    while j < len(b):
        pole.append(b[j])
        j += 1
    return pole
def vypis_a_uloz_pole(c, nazov_suboru):
    subor = open(nazov_suboru, "w")
    for slovo in c:
        print(slovo)
        print(slovo, file = subor)
    subor.close()

a = nacitaj_pole("slovnik1.txt")
b = nacitaj_pole("slovnik2.txt")
print("Pole a = ", a)
print("Pole b = ", b)
c = zluc_polia(a, b)
print("Zlúčené pole =", c)
vypis_a_uloz_pole(c,"slovnik.txt")
```

46) 3.10i Zobrazovať prvky poľa vypísaním alebo vykreslením (napríklad: stĺpcový graf).

❖ Zostavte program na zobrazenie stĺpcového grafu. Program načíta zo vstupu výšky jednotlivých stĺpcov. Zadávanie sa ukončí zadaním záporného čísla. Vykreslené výšky stĺpcov budú prispôsobené tak, aby najvyšší stĺpec mal konštantnú výšku napr. 200. Pod každým stĺpcom bude napísaná jeho hodnota.

```
import tkinter

canvas = tkinter.Canvas()

#kreslenie jedného stĺpca
def stlpec(hodnota, max, poradie):
    vyska = hodnota / max * 200
    canvas.create_rectangle(poradie * 20, 200, 20 + poradie * 20, 200 -
vyska, fill="blue")
    canvas.create_text(10 + poradie * 20, 220,text=str(hodnota))

max = 0
pole = []

# nacitanie vysok stlpcov
while True:
    vyska = int(input("Zadaj výšku stĺpca: "))
    if vyska < 0:
        break
    pole.append(vyska)
    if vyska > max:
        max = vyska

#vykreslenie stĺpcov
canvas.pack()
i = 1
for vyska in pole:
    stlpec(vyska, max, i)
    i += 1

canvas.mainloop()
```

Tip na riešenie:

☞ Načítané výšky stĺpcov uchovajte v poli.

☞ Pamätajte si najväčšiu výšku stĺpca `max`. Tento stĺpec bude mať výšku v pixeloch 200.

☞ Pri vykresľovaní stĺpca bude jeho výška v pixeloch $= \frac{VýškaStĺpca}{max} . 200$ (trojčlenka)

47) 3.11a Riešiť problémy, v ktorých sa využíva textový súbor: zobraziť obsah súboru (napríklad: kreslenie zakódovaného obrázka, naplnenie poľa hodnotami).

Zadanie:

- Zostavte program, ktorý v textovom režime vykreslí zakódovaný obrázok zložený z hviezdičiek. Návod na nakreslenie bude zapísaný v textovom súbore takto:

  ```
  počet_hviezdičiek počet_medzier počet_hviezdičiek ... -1 ... počet_hviezdičiek
  počet_medzier počet_hviezdičiek ... -1
  počet_hviezdičiek počet_medzier počet_hviezdičiek ... -1
  ```

  Napr. postupnosť čísel: 6 -1 0 1 3 1 1 -1 0 2 4 -1 vykreslí:

  ```
  ******
   *** *
    ****
  ```

```
subor = open("zakodovany_obrazok.txt", "r")
riadok = subor.readline()
znak = "*" #ktory znak sa bude vykreslovat
while riadok != "":

    cislo = int(riadok)
    if cislo == -1:
        print()
        znak ="*" # v novom riadku začíname kreslením hviezdičky
    else:
        print(znak * cislo, end = "") #vykreslenie znakov
        if znak == "*": #zmena znaku - ak sa kreslila hviezdička, bude sa
kresliť medzera
            znak = " "
        else:
            znak = "*"

    riadok = subor.readline()

subor.close()
```

Tip na riešenie:

- ☞ Kvôli uľahčeniu načítavania čísel zo súboru môže byť každé číslo na samostatnom riadku.

48) 3.11b Riešiť problémy, v ktorých sa využíva textový súbor: používať textový súbor ako vstup alebo výstup (napríklad: čítanie - vstupných údajov pre výpočty; zapisovanie výsledkov výpočtov, obsahu polí a pod.).

Zadanie:

- Vytvorte program, ktorý bude počítať výsledný prospech na vysvedčení. Vstupné údaje – známky načíta zo vstupného súboru. Výsledné údaje: známky a prospech vypíše na obrazovku.

  Príklad:

  Vstupný súbor:

  1

  1

  1

  2

  Výstup na obrazovku:

  1, 1, 1, 2, – prospel s vyznamenaním

  Pozn: priemer <= 1,5 – prospel s vyznamenaním
  priemer <= 2 prospel veľmi dobre
  priemer > 2 prospel
  ak je medzi známkami 5 - neprospel

```
def vypocitaj_prospech(priemer, je_patka):
    if je_patka:
        return "Neprospel"
    elif priemer <= 1.5:
        return "Prospel s vyznamenaním"
    elif priemer <= 2:
        return "Prospel veľmi dobre"
    else:
        return "Prospel"

vstupny_subor = open("znamky.txt", "r")

riadok = vstupny_subor.readline()
sucet = 0
pocet = 0
je_patka = False

while riadok != "":
    znamka = int(riadok)
    print(znamka, end=", ")
    pocet += 1
    sucet += znamka

    if znamka == 5:
        je_patka = True

    riadok = vstupny_subor.readline()

prospech = vypocitaj_prospech(sucet/pocet, je_patka)
print(prospech)

vstupny_subor.close()
```

49) 3.11c Zisťovať štatistické údaje o obsahu textového súboru (napríklad: počet riadkov súboru, počet slov, počet znakov, súčet čísel a pod.).

Zadanie:

- Zostavte program, ktorý analyzuje obsah vstupného súboru. Zistí počet slov a vypíše najdlhšie slovo. Slová sú oddelené medzerou, bodkou, čiarkou alebo znakom nového riadka. Ak je najdlhších slov viac, vypíše prvé v poradí.

```
subor = open("text.txt", "r")
cely_subor = subor.read() + " "

pocet_slov = 0
pocet_pismen_v_slove = 0
najdlhsie_slovo = ""
citam_slovo = False
oddelovace= (" ","\n", ",", ".","\t") #oddelovace konca slov
slovo = ""
for znak in cely_subor:
    if citam_slovo: #prave sa nacitavaju znaky slova
        if znak in oddelovace: #nasiel sa koniec slova, jeden z oddelovacov
            print(slovo)
            if len(slovo) > len(najdlhsie_slovo):
                najdlhsie_slovo = slovo
            citam_slovo = False #nastavenie sa do modu citania oddelovacov
        else:
            slovo += znak
    else: #prave sa nacitavaju sa oddelovace
        if not znak in oddelovace:
            slovo = znak #nasiel sa zaciatok noveho slova
            pocet_slov += 1
            citam_slovo = True #nastavenie sa do modu citania slovs

print("Počet slov je", pocet_slov)
print("Najdlhšie slovo je", najdlhsie_slovo)

subor.close()
```

Tip na riešenie:

- Naše riešenie je založené na tom, že sa čítajú postupne všetky znaky vstupného lisúboru.
- Slová môžu byž oddelené oddeľovačom: medzera, čiarka, bodka, nový riadok…
- Čítanie znakov môže byť v dvoch stavoch:
    1. „číta sa slovo“ : postupne sa čítajú znaky a slovo končí nájdením oddeľovača. Vtedy sa prepne do stavu „číta sa oddeľovač“
    2. „číta sa oddeľovač“ : postupne sa čítajú znaky a keď sa nájde písmeno, prepne sa do stavu „číta sa slovo“

50) 3.11d Pracovať súčasne s viacerými textovými súbormi (napríklad: kopírovanie súborov, zlúčenie dvoch súborov, kopírovanie s filtrovaním a pod.).

Zadanie:

- ❖ Zostavte program, ktorý skopíruje vstupný súbor do výstupného, pričom odstráni zo vstupného súboru všetky čísla a veľké písmená a nahradí ich podtržítkom.

```
vstupny_subor = open("vstup50.txt", "r")
vystupny_subor = open("vystup50.txt", "w")

cely_subor = vstupny_subor.read()
for znak in cely_subor:
    if znak >= "A" and znak <= "Z" or znak >= "0" and znak <= "9" or znak
in "ÁČĎÉĚÍĽĽŇÓŘŔŠŤÚŽÝ":
        znak ="_"

    vystupny_subor.write(znak)

vstupny_subor.close()
vystupny_subor.close()
```

Tip na riešenie:

- ☞ V našom riešení prechádzame v cykle celý súbor po znakoch.
- ☞ Testujeme či znak je veľké písmeno, číslica, alebo veľké písmeno s diakritikou. Ak takýto znak nájdeme do výstupného súboru zapíšeme podtržítko, ak nie, zapíšeme pôvodný znak.