Štruktúrované údajové typy

Program môže spracúvať celé (typ int) alebo reálne (typ float) čísla, logické hodnoty False alebo True (typ bool), reťazce znakov (typ str) a iné údajové typy (tuple, list,...). Údajové typy int, float a bool nazývame jednoduché, str je štruktúrovaný údajový typ, pretože má štruktúru. Hodnoty typu str – reťazce, sa skladajú zo znakov, sú tvorené nemennou postupnosťou znakov. Krátke zopakovanie: Reťazec je nemenná[1] (immutable) postupnosť znakov Unicode údajového typu str. Hodnoty typu str sú uzavreté v úvodzovkách alebo apostrofoch. Prázdny reťazec je reťazec, ktorý nemá medzi ohraničením ani jeden znak. K jednotlivým znakom v reťazci sa dá dostať pomocou výrazu reťazec[index], pričom prvý znak v reťazci má index 0. Postupne ku všetkým znakom v reťazci sa dá dostať napríklad zápismi: `for` znak `in` reťazec: ...spracovanie znaku znak... alebo `for` index `in` range(len(reťazec)): ...spracovanie znaku reťazec[index]....

Ďalším štruktúrovaným údajovým typom je tuple – n-tica.

N-tica je postupnosť ľubovoľných nemenných (immutable) hodnôt uzavretá v okrúhlych zátvorkách. Hodnoty n-tice sú oddelené čiarkami a odporúča sa celú postupnosť uzavrieť do okrúhlych zátvoriek. Prázdna n-tica sa zapisuje (), jednoprvková (hodnota,) - čiarka je povinná, aby bolo jednoznačne povedané, že ide o n-ticu a nie zbytočne použité zátvorky.

Najpoužívanejšie funkcie a operácie:

len()	- vráti počet prvkov n-tice
prvok in n-tica	- vráti True, ak sa prvok v n-tici nachádza, inak vráti False (opačne not in)
+	- spojí (zreťazí) kópie dvoch n-tíc
*n	- n-krát zreťazí kópie n-tice
n-tica[od:po:krok]	- rezací operátor
<, <=, ==, !=, >=,>	- porovnanie dvoch n-tíc, porovnávanie sa aplikuje na jednotlivé prvky v poradí, ako sú uvedené (musia byť porovnateľné; nemožno napríklad typ int porovnať s typom str!)
+= alebo *=	- rozšíria enticu

Ako vidieť z ukážky, použitie zátvoriek v určitých situáciách nie je nevyhnutné, postupnosť `"Fero", True, -4, 5.28` bola pochopená ako tuple, ale sú situácie, v ktorých bez použitia zátvoriek „nebudeme správne pochopení“.

Ukážky:

```
>>> en = ()                                    # vytvorenie prázdnej n-tice
>>> for prvok in range(1,10):                  # naplnenie n-tice číslami 1 až 9
      en += (prvok,)
>>> print(en)
(1, 2, 3, 4, 5, 6, 7, 8, 9)
>>> en += (10,11,12,13)                        # rozšírenie n-tice (pridanie hodnôt na koniec)
>>> print(en)
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13)
>>> en = (0,) + en                             # pridanie nuly na začiatok n-tice
>>> print(en)
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13)
>>> en = en[1:11]                              # použitie rezacieho operátora
>>> print(en)
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
>>> en1 = 5*(1,2)                              # použitie opakovacieho operátora
>>> print(en1)
(1, 2, 1, 2, 1, 2, 1, 2, 1, 2)
>>> 0 in en                                    # použitie operátora príslušnosti in
False
>>> 1 in en
True
>>> en = "Fero", True, -4, 5.28                # vytvorenie n-tice vymenovaním prvkov
>>> en
('Fero', True, -4, 5.28)
```

[1] Pri pokuse zmeniť hodnotu niektorého prvku reťazca sa vypíše chybové hlásenie: TypeError: 'str' object does not support item assignment - Chyba typu: objekt 'str' nepodporuje priradenie položky.

```
>>> en[1] = False                                  # nedovolená operácia - zmena hodnoty prvku n-tice
Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    en[1] = False
TypeError: 'tuple' object does not support item assignment

>>> ("Pavol",1,False) == ("Pavol",1,False)        # porovnávanie n-tíc
True
>>> ("Pavol",1,False) != ("Pavol",1,False)        # porovnávané hodnoty, t.j. prvá s prvou, druhá
False                                             # s druhou atď., v porovnávaných n-ticiach  musia byť
>>> ("Pavol",1,False) < ("Pavol",1,True)          # rovnakého typu!
True
>>> ("Pavol",2,False) < ("Pavol",1,True)
False
>>> ("Pavol",2,False) < ("Peter",1,True)
True
>>> ("Pavol",2,False) < ("Pavol",1,True)
False
>>> ("Pavol",1) < ("Pavol",1,False)               # na konci rozšírená n-tica je „neskôr“ ako pôvodná
True                                              # n-tica
>>> ("Pavol",1,False) < ("Pavol",1)
False
```

Príklad T.1

Vytvorte program, tzv. mincovku. Po zadaní sumy - celé nezáporné číslo, program vypíše minimálny počet bankoviek a mincí potrebných na vyplatenie zadanej sumy. Napríklad pre sumu 10 018 program vypíše:

```
Mincovka pre sumu: 10018
 5000 € ....  2
 1000 € ....  0
  500 € ....  0
  200 € ....  0
  100 € ....  0
   50 € ....  0
   20 € ....  0
   10 € ....  1
    5 € ....  1
    2 € ....  1
    1 € ....  1
```

Riešenie 1 - dve samostatné funkcie pre vyvezenie počtu bankoviek alebo mincí zadanej hodnoty a novej sumy:

```
HODNOTY_MENY = (5000,1000,500,200,100,50,20,10,5,2,1)                    # použitie n-tice

def vratPocetZadanejHodnoty(suma, hodnota):
    return suma//hodnota

def vratNovuSumu(suma,hodnota):
    return suma%hodnota

def mincovka(suma):
    for hodnota in HODNOTY_MENY:
        print("{:5d} € .... {:2d}".format(hodnota, vratPocetZadanejHodnoty(suma,hodnota)))
        suma = vratNovuSumu(suma,hodnota)

# ============== HLAVNÝ PROGRAM - Použitie funkcií ==============
suma = int(input("Mincovka pre sumu: "))
mincovka(suma)
```

Riešenie 2 - vyvezenie počtu bankoviek alebo mincí zadanej hodnoty a novej sumy z jednej funkcie:

```
HODNOTY_MENY = (5000,1000,500,200,100,50,20,10,5,2,1)

def vratPocetZadanejHodnoty(suma, hodnota):
    return suma//hodnota, suma%hodnota
```

```
def mincovka(suma):
    for hodnota in HODNOTY_MENY:
        pocet, suma = vratPocetZadanejHodnoty(suma, hodnota)
        print("{:5d} € .... {:2d}".format(hodnota, pocet))

# ============== HLAVNÝ PROGRAM - Použitie funkcií ==============
suma = int(input("Mincovka pre sumu: "))
mincovka(suma)
```

Riešenie 3 - suma ako globálna premenná:

```
HODNOTY_MENY = (5000,1000,500,200,100,50,20,10,5,2,1)

def vratPocetZadanejHodnoty(hodnota):
    global suma
    pocet = suma//hodnota
    suma %= hodnota
    return pocet

def mincovka():
    global suma
    for hodnota in HODNOTY_MENY:
        print("{:5d} € .... {:2d}".format(hodnota, vratPocetZadanejHodnoty(hodnota)))

# ================== HLAVNÝ PROGRAM ==================
suma = int(input("Mincovka pre sumu: "))
mincovka()
```

Ďalším štruktúrovaným údajovým typom je list – zoznam.

Najjednoduchšie je pre začiatok údajový typ list definovať ako postupnosť ľubovoľných hodnôt uzavretú v hranatých zátvorkách. Hodnoty zoznamu sú oddelené čiarkami a možno ich meniť!

Ukážky:

```
>>> zoz = []                                    # vytvorený prázdny zoznam
>>> zoz = ["Fero", -4, 13.58, False]            # zoznam vytvorený vymenovaním prvkov
>>> zoz
['Fero', -4, 13.58, False]
>>> zoz[-1] = True
>>> zoz
['Fero', -4, 13.58, True]
>>> zoz = [cislo for cislo in range(1,10)]      # zoznam vytvorený vygenerovaním prvkov
>>> zoz
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> zoz = [0] + zoz + [10]
>>> zoz
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> zoz = zoz[1:]
>>> zoz
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
zoz = list(cislo for cislo in range(1,10,2))
>>> zoz
[1, 3, 5, 7, 9]
zoz = 10*[True]                                 # zoznam vytvorený opakovacím operátorom
>>> zoz
[True, True, True, True, True, True, True, True, True, True]
>>> zoz = ["GP" "Nitra" for _ in range(5)]
>>> zoz
['GPNitra', 'GPNitra', 'GPNitra', 'GPNitra', 'GPNitra']
```

Operátory

zoz[i] = x	položka s indexom i je nahradená hodnotou x
del zoz[i]	odstráni prvok zoznamu na indexe i (dôjde k preindexovaniu poľa!)
zoz[i:j] = v	úsek zoznamu s indexmi i až j-1 je nahradený zoznamom v

del zoz[i:j]	odstránenie úseku zoznamu (rovnako ako zoz[i:j] = [])
x in zoz	test príslušnosti (je x prvkom zoznamu?)
x not in zoz	negácia testu príslušnosti
zoz1 + zoz2	spojí zoznamy
k*zoz, zoz*k	k opakovaní prvkov zoznamu (k je typu int)
zoz[i]	vráti kópiu i-teho prvku
zoz[i:j]	vráti kópiu časti zoznamu od i-teho po j-1 index zoznamu
for x in zoz	prechádzanie prvkami zoznamu

Užitočné interné funkcie a metódy :

len(zoz)	vráti dĺžku zoznamu
min(zoz)	vráti najmenší prvok zoznamu
max(zoz)	vráti najväčší prvok zoznamu
sum(zoz)	súčet prvkov číselného zoznamu
zoz.append(x)	pridá hodnotu x na koniec zoznamu (analogicky: zoznam += x)
zoz1.extend(zoz2)	rozšíri zoznam zoz1 o prvky zoznamu zoz2
zoz.insert(i, x)	vloží hodnotu x do zoznamu na index i (dôjde k preindexovaniu poľa!)
zoz.pop(i)	odstráni prvok zoznamu na indexe i a vráti tento prvok ako hodnotu
zoz.pop()	odstráni posledný prvok zoznamu a vráti tento prvok ako hodnotu
zoz.remove(x)	odstráni zo zoznamu prvý výskyt hodnoty x (ak sa x v zozname nevyskytlo, vráti chybu)
zoz.sort()	utriedi vzostupne zoznam (vráti None)
zoz.reverse()	otočí poradie prvkov v zozname
zoz.index(x)	vráti index prvého výskytu hodnoty x v zozname (ak x v zozname nie je, vráti chybu)
zoz.count(x)	vráti počet výskytov hodnoty x v zozname
sorted(zoznam)	vráti vzostupne utriedený zoznam

POZOR!
Pri odstraňovaní prvkov zo zoznamu dochádza automaticky k preindexovaniu zoznamu, t.j. ak sme odstránili prvok zoznam[i], na jeho miesto bude umiestnený prvok zoznam[i+1] (ak existuje) a bude mať index i! Preto v úlohách s viacnásobným odstraňovaním prvku zo zoznamu nie je veľmi vhodné použiť for-cyklus ale while-cyklus, v ktorom index i zvýšime o 1 až keď už nedôjde k odstráneniu prvku na indexe i (inak bude nový prvok na indexe i neotestovaný na odstránenie). Na problém s meniacimi sa indexmi treba myslieť aj pri vkladaní hodnôt do zoznamu príkazom zoznam.insert(index, hodnota)!

Vo viacerých programovacích jazykoch sa vyskytujú údajové typy s podobnými vlastnosťami ako v Pythone pomenovaný údajový typ list (zoznam). Jedným z často používaných názvov je **pole** (array). Aby ste si osvojili aj tento pojem, budeme často namiesto názvu zoznam používať pre údajový typ list názov pole. V závere úloh s údajovým typom list prejdeme celkom na označenie zoznam.
Keďže problematika je pomerne rozsiahla, rozdelíme ju na viacej študijných textov a to:

- Štruktúrované údajové typy (ten práve čítate)
- Pole - vytvorenie a vypísanie
- Lineárne vyhľadávanie
- Triedenie
- Binárne vyhľadávanie
- Zásobník a rad

Údajový typ set - množina

Množina (set) je neusporiadaná skupina neopakujúcich sa hodnôt nemeniteľného údajového typu.
„Neusporiadaná“ skupina znamená, že hodnota bude pridaná svojvoľne na ľubovoľné miesto do množiny; „neopakujúcich sa“ hodnôt znamená, že ak zadáme príkaz na pridanie hodnoty do množiny, bude táto hodnota vložená do množiny, len ak sa v nej nevyskytuje a „nemeniteľného“ znamená immutable údajového typu (pozri úvod tohto študijného textu).

Množiny v Pythone sú množinami v matematickom zmysle slova, takže s nimi môžeme vykonávať štandardné množinové operácie, ako prienik (&), zjednotenie (|) a rozdiel (-) dvoch množín. Množinami sa nebudeme podrobnejšie zaoberať pre ich špecifické vlastnosti a tým aj oblasti použitia, preto uvedieme len ukážku základných vlastností a príklad možného použitia.

```
>>> mnozina = set()      # vytvorenie prázdnej množiny
>>> set()                # vypísanie množiny
set()                    # prázdna množina!
>>> type(set())          # príkaz na vypísanie, akého údajového typu je set()
<class 'set'>
>>> mnozina = {}         # takto NEMOŽNO vytvoriť prázdnu množinu (vytvorí sa iný údajový typ - slovník)!
>>> mnozina
{}
>>> type(mnozina)
<class 'dict'>           # Python pochopil ako vytvorenie prázdneho slovníka (dictionary)
>>> mnozina = {7}        # vytvorenie jednoprvkovej množiny

>>> mnozina                          # vypísanie množiny
{7}
>>> type(mnozina)                    # kontrola, či ide o typ množina
<class 'set'>
>>> len(mnozina)                     # vypísanie počtu prvkov množiny
1
>>> mnozina.add("Peter")             # pridanie (immutable) hodnoty do množiny
>>> mnozina
{7, 'Peter'}
>>> mnozina.add(True)
>>> mnozina.add(False)
>>> mnozina
{False, True, 7, 'Peter'}
>>> mnozina.add("Pavol")
>>> len(mnozina)
5
>>> mnozina.add(7)                   # hodnota, ktorá sa už množine vyskytuje, NEBUDE PRIDANÁ!
>>> mnozina
{False, True, 'Pavol', 7, 'Peter'}
>>> mnozina.discard(7)               # odstránenie hodnoty z množiny
>>> mnozina
{False, True, 'Pavol', 'Peter'}
>>> "Pavol" in mnozina               # operátor príslušnosti (či množina obsahuje hodnotu)
True
>>> mnozina
{False, True, 'Pavol', 'Peter'}
>>> mnozina.add(0)                   # pozor na totožnosť False - 0 a True - 1!
>>> mnozina
{False, True, 'Pavol', 'Peter'}
>>> mnozina .discard(False)
>>> mnozina
{True, 'Pavol', 'Peter'}
>>> mnozina.add(0)
>>> mnozina
{0, True, 'Pavol', 'Peter'}
>>> mnozina.add(1)
>>> mnozina
{0, True, 'Pavol', 'Peter'}
>>> mnozina.add(2)
>>> mnozina
{0, True, 2, 'Pavol', 'Peter'}
>>> mnozina.clear()                  # vymazanie prvkov množiny
>>> mnozina
set()                                # prázdna množina
>>> mnozina = {"Ada","Zora"}
```

```
>>> mnozina
{'Ada', 'Zora'}
>>> mnozina = mnozina | {"Adela","Zuzana","Zora"}     # zjednotenie dvoch množín
>>> mnozina
{'Ada', 'Zuzana', 'Zora', 'Adela'}
```

Príklad S.1
Vytvorte funkciu, ktorá odstráni všetky opakujúce sa hodnoty zo zoznamu (poľa).

Elegantným riešením je „pretypovanie“ list-u (zoznamu) na set (množinu), v ktorej sú automaticky odstránené duplicitné hodnoty a následne „pretypovanie“ späť na typ zoznam.

```
import pole

def odstranOpakujuceSaPrvky(pole):
    pom = set(pole)
    pole = list(pom)
    return pole               # jediným príkazom funkcie odstran...: return list(set(pole))
```

Použitie:

```
poleInt = pole.vytvorIntNahodne()
poleInt = odstranOpakujuceSaPrvky(poleInt)
pole.vypis(poleInt, "Pole bez opakujúcich sa hodnôt:")
```

Výpis:

```
Počet prvkov poľa: 13
Prvky poľa:
[1, 1, 5, 6, 5, 8, 1, 8, 6, 8, 5, 1, 8]
Pole bez opakujúcich sa hodnôt:
[8, 1, 5, 6]
```

Použitie bez definovania funkcie `odstranOpakujuceSaPrvky()`:

```
poleInt = pole.vytvorIntNahodne()
poleInt = list(set(poleInt))
pole.vypis(poleInt, "Pole bez opakujúcich sa hodnôt:")
```

Údajový typ slovník - dict

Slovník (dictionary) je neusporiadaná kolekcia (skupina) dvojíc kľúč : hodnota, oddelených čiarkami, uzavretá v zložených zátvorkách. Kľúčom môže byť ľubovoľný nemeniteľný typ, napríklad celé čísla, reťazce ale aj n-tice. Keď do slovníka pridáme kľúč, musíme do neho súčasne pridať aj hodnotu, ktorá ku kľúču patrí (hodnotu môžeme v budúcnosti kedykoľvek zmeniť). Hodnoty v slovníku môžu byť ľubovoľného dátového typu vrátane čísel, hodnôt False, True, reťazcov, zoznamov alebo dokonca slovníkov. Pythonovské slovníky sú optimalizované pre získavanie hodnoty k zadanému kľúču, ale nie naopak (kľúč nemôžeme triviálne – bez hľadania, získať na základe poznania hodnoty).

```
>>> slovnik = {}                        # vytvorenie prázdneho slovníka
>>> slovnik                             # obsah slovníka (položky slovníka)
{}
```

Vytvorenie slovníka zadaním položiek

```
>>> slovnik = {"Karol":1.25, "Tomáš":2.78, "Dana": 1.00, "Alexandra": 1.11}
>>> slovnik                             # nový obsah slovníka (položky slovníka)
{'Alexandra': 1.11, 'Tomáš': 2.78, 'Dana': 1.0, 'Karol': 1.25}
>>> slovnik["Dana"]                     # získanie hodnoty prislúchajúcej ku kľúču "Dana"
1.0
>>> slovnik["Zuzana"] = 2.34            # pridanie položky "Zuzana" : 2.34 do slovníka
>>> slovnik
{'Alexandra': 1.11, 'Tomáš': 2.78, 'Dana': 1.0, 'Zuzana': 2.34, 'Karol': 1.25}
>>> len(slovnik)                        # počet položiek v slovníku
5
>>> del slovnik["Alexandra"]            # odstránenie položky s kľúčom „Alexandra“ zo slovníka
>>> slovnik
{'Tomáš': 2.78, 'Dana': 1.0, 'Zuzana': 2.34, 'Karol': 1.25}
```

V slovníku sa nemôžu nachádzať duplicitné kľúče. Ak priradíme hodnotu k už existujúcemu kľúču v slovníku, dôjde k prepísaniu pôvodnej hodnoty.

```
>>> slovnik["Zuzana"] = 1.00        # zmena (prepísanie) hodnoty kľúča "Zuzana"
>>> slovnik
{'Alexandra': 1.11, 'Tomáš': 2.78, 'Dana': 1.0, 'Zuzana': 1.0, 'Karol': 1.25}
```

Najpoužívanejšie metódy

operátor in	vráti True, ak sa kľúč nachádza v slovníku, inak False
len(slovnik)	vráti počet položiek v slovníku
slovnik.keys()	vráti zoznam kľúčov v slovníku
slovnik.values()	vráti všetky hodnoty kľúčov zo slovníka
slovnik.items()	vráti všetky dvojice kľúč, hodnota slovníka
del slovnik[kľúč]	odstráni položku so zadaným kľúčom zo slovníka
slovnik.clear()	odstráni všetky položky zo slovníka

Ukážky:

```
>>> slovnik = {'Tomáš': 2.78, 'Dana': 1.0, 'Zuzana': 1.0, 'Karol': 1.25}
>>> for kluc, hodnota in slovnik.items():
        print("{:10} {:5.2f}".format(kluc, hodnota))

Dana        1.00
Zuzana      1.00
Karol       1.25
Tomáš       2.78
```

Vypísanie položiek slovníka utriedených podľa kľúča:

```
>>> for kluc in sorted(slovnik):
        print("{:10} {:5.2f}".format(kluc, slovnik[kluc]))

Dana        1.00
Karol       1.25
Tomáš       2.78
Zuzana      1.00
```

Vypísanie kľúčov zviazaných so zadanou hodnotou:

```
>>> hladat = 1.00
>>> for (kluc, hodnota) in slovnik.items():
      if hodnota == hladat:
          print(kluc)

Dana
Zuzana
```

Utriedenie položiek slovníka podľa kľúča - uloží do zoznamu!

```
>>> slovnik
{'Adela': 2.47, 'Zuzana': 1.0, 'Zora': 3.41, 'Ada': 3.41, 'Dana': 1.0, 'Tomáš': 2.78}
>>> zoznam_utried_kluc = sorted(slovnik.items())
```

alebo

```
>>> zoznam_utried_kluc = sorted(slovnik.items(), key=lambda podla: podla[0])
>>> zoznam_utried_kluc
[('Ada', 3.41), ('Adela', 2.47), ('Dana', 1.0), ('Tomáš', 2.78), ('Zora', 3.41), ('Zuzana', 1.0)]
>>> type(zoznam_utried_kluc)
<class 'list'>
```

Utriedenie položiek slovníka podľa hodnoty - uloží do zoznamu!

```
>>> zoznam_utried_hodnota = sorted(slovnik.items(), key=lambda podla: podla[1])
>>> zoznam_utried_hodnota
[('Zuzana', 1.0), ('Dana', 1.0), ('Adela', 2.47), ('Tomáš', 2.78), ('Zora', 3.41), ('Ada', 3.41)]
```

Krajší výpis položiek slovníka utriedených podľa hodnoty (teda krajší výpis zoznamu obsahujúceho dvojice - prvky typu tuple):

```
for prvok in zoznam_utried_hodnota:
    print("{:10}{:5.2f}".format(prvok[0],prvok[1]))
```

Výpis:

```
Dana       1.00
Zuzana     1.00
Karol      1.25
Tomáš      2.78
```

Príklad S.1

Vytvorte program, ktorý po zadaní textu vypíše tabuľku: znak a počet jeho výskytov v texte.

```
def vratPoctyZnakov(text):
    pocty = {}
    for znak in text:
        if znak in pocty:                # ak sa už znak (kľúč) v slovníku vyskytuje
            pocty[znak] += 1             # tak zväčši počet jeho výskytov o 1
        else:
            pocty[znak] = 1              # inak vlož do slovníka nový kľúč (znak) s hodnotou 1
    return pocty

def vypisSlovnik(slovnik):
    for (kluc, hodnota) in slovnik.items():
        print(kluc, hodnota)

def main():                              # definícia funkcie, ktorá "spúšťa" program
    text = input("Zadaj text: ")
    poctyZnakov = vratPoctyZnakov(text)
    vypisSlovnik(poctyZnakov)

# ========================================
main()
```

Program doplnený o výpis utriedený podľa Unicodu (kľúča):

```
def vypisUtriedSlovnikKluc(slovnik):
    for kluc in sorted(slovnik):
        print(kluc, slovnik[kluc])
```

Program doplnený o výpis utriedený zostupne podľa počtu výskytov jednotlivých znakov:

```
def vypisUtriedSlovnikHodnota(slovnik):
    zoznam = sorted(slovnik.items(), key=lambda podla: podla[1], reverse=True)
    for prvok in zoznam:
        print(prvok[0],prvok[1])
```

Doplnená funkcia main():

```
def main():
    text = input("Zadaj text: ")
    if text == "":
        text = "Abraka dabraaaaa!"
        print(text)
    poctyZnakov = vratPoctyZnakov(text)
    vypisSlovnik(poctyZnakov)
    print("Utriedené podľa Unicode:")
    vypisUtriedSlovnikKluc(poctyZnakov)
    print("Utriedené podľa počtu výskytov:")
    vypisUtriedSlovnikHodnota(poctyZnakov)
```

Príklad S.2
Vytvorte funkciu, ktorá k ľubovoľnému slovníku (typ dict) vytvorí opačný slovník (kľúč sa stane hodnotou a hodnota kľúčom). Predpokladajte, že v pôvodnom slovníku sa nevyskytujú dve rovnaké hodnoty.

```
def otocSlovnik(slovnik):
    o_slovnik = {}
    duplicitne_kluce = []
    for kluc, hodnota in slovnik.items():
        if hodnota in o_slovnik:
            duplicitne_kluce.append(hodnota)
        else:
            o_slovnik[hodnota] = kluc
    print("Duplicitné kľúče:", duplicitne_kluce)
    return o_slovnik

slovnik = {0:"nula", 1:"jedna", 2:"dva", 3:"tri", 4:"štyri", 5:"štyri"}
novy = otocSlovnik(slovnik)
print("Nový (otočený) slovník:", novy)
```

Príklad S.3
Vytvorte jednoduchý slovensko - anglický prekladový slovník. Ponuka nech umožňuje hľadať v slovníku anglický ekvivalent po zadaní slovenského slova. Ak sa slovenské slovo v slovníku nenachádza, nech ho umožní pridať do slovníka aj s anglickým prekladom.
Program doplňte tak, aby slovník umožňoval obojsmerný preklad.

```
slovnikSA = {"začať"    :"begin",
             "ak"       :"if",
             "koniec"   :"end",
             "pokiaľ"   :"while",
             "pre"      :"for",
             "opakovať" :"repeat",
             "dostať"   :"get",
             "vrátiť"   :"return",
             "nastaviť" :"set" }

def main():
    while True:
        print()
        print("Preklad slovensko -> anglický ..... 0")
        print("Preklad anglicko -> slovenský ..... +")
        print("Koniec ............................. Enter")
        odpoved = input("Tvoja voľba: ")
        if odpoved == "":
            break
        if odpoved == "0":
            sl_slovo = input("Preložiť slovenské slovo: ")
            if sl_slovo in slovnikSA:
                print(sl_slovo, " - ", slovnikSA[sl_slovo])
            else:
                print(sl_slovo, " -  ?")
                an_slovo = input("Preklad: ")
                slovnikSA[sl_slovo] = an_slovo
        elif odpoved == "+":
            hladat = input("Preložiť anglické slovo: ")
            naslo_sa = False
            for sl_slovo, an_slovo in slovnikSA.items():
                if an_slovo == hladat:
                    print(an_slovo, " - ", sl_slovo)
                    naslo_sa = True
                    break
            if not naslo_sa:
                print(hladat, " -  ?")
                sl_slovo = input("Preklad: ")
                slovnikSA[sl_slovo] = hladat
```

```
        else:
            print("Stlačený zlý kláves!")
        ''' iné riešenie:
            an_slovo = input("Preložiť anglické slovo: ")
            naslo_sa = False
            for sl_slovo in slovnikSA.keys():
                if slovnikSA[sl_slovo] == an_slovo:
                    print(an_slovo, " - ", sl_slovo)
                    naslo_sa = True
                    break
            if not naslo_sa:
                print(an_slovo, " -  ?")
                sl_slovo = input("Preklad: ")
                slovnikSA[sl_slovo] = an_slovo
        else:
            print("Stlačený zlý kláves!")
        '''
main()
```

Ponuku programu doplňte o možnosť „inteligentného“ vypísania dvojíc slov v slovníku.

Príklad S.4
Vytvorte program, ktorý po zadaní bodov v rovine so súradnicami x a y vypočíta obsah minimálneho obdĺžnika, obsahujúceho všetky zadané body. Strany obdĺžnika sú rovnobežné so súradnicovými osami.

```
def vratMinMaxSuradnice(body):
    prvy = True
    for ozn, sur in body.items():
        if prvy:                                    # priradenie počiatočných hodnôt
            minX = maxX = sur[0]
            minY = maxY = sur[1]
            prvy = False
        else:
            if sur[0] < minX: minX = sur[0]
            if sur[0] > maxX: maxX = sur[0]
            if sur[1] < minY: minY = sur[1]
            if sur[1] > maxY: maxY = sur[1]
    return minX, minY, maxX, maxY                   # funkcia vráti n-ticu!

def vratMinObsah(minMaxSur):
    return (minMaxSur[2] - minMaxSur[0])*(minMaxSur[3] - minMaxSur[1])
def main():
    body = {"A":[10,0],"B":[-4,8],"C":[5,2],"D":[9,4],"E":[5,-3],"F":[-10,-10],"G":[0,10]}
    print("Štvorica (minX, minY, maxX, maxY):", vratMinMaxSuradnice(body))
    print("Obsah minimálneho obdĺžnika je {:.1f} j\u00b2."
                                    .format(vratMinObsah(vratMinMaxSuradnice(body))))
main()
```

Predpokladajme, že bod s názvom A sa vždy vyskytuje v dict body. Viete zabezpečiť priradenie počiatočných hodnôt vo funkcii `vratMinMaxSuradnice(body)` aj iným spôsobom?
Úlohu riešte v priestore (tri súradnice) a hľadajte minimálny kváder obsahujúci všetky zadané body.

Príklad S.5
Vytvorte program, ktorý prečíta z databázy meno pracovníka a počet jeho odpracovaných hodín za mesiac a vypočíta jeho mzdu. Hodinová mzda je 5€. Na záver program vypíše celkovú vyplatenú sumu.
Požadovaný výpis:

```
Pracovník      Mzda v €
Kováčová         840.00
Novák            937.50
Malý            1000.00
======================
Súčet           3677.50
```

Príklad S.6
Databáza obsahuje záznamy: meno, hmotnosť v kg a výšku v m. Program nech vypíše tabuľku podľa ukážky: meno a k nemu prislúchajúce BMI a hodnotenie. BMI sa počíta ako hmotnosť/výška^2. Hodnotenie podvýživa sa vypíše, ak bmi je menšie ako 18, v norme sa vypíše, ak bmi je menšie ako 22, inak sa vypíše nadváha.
Ukážka:

```
Meno       BMI   Hodnotenie
Peter      17.43 podvýživa
Adela      18.42 v norme
Pavol      25.76 nadváha
Zuzana     24.09 nadváha
```

```
def bmi(hmotnost, vyska):
    return hmotnost/vyska/vyska

def hodnotenie(bmi):
    if bmi < 18:
        return "podvýživa"
    elif bmi < 22:
        return "v norme"
    else:
        return "nadváha"

def main(databaza):
    print("{:10} {:5} {}".format("Meno", "BMI", "Hodnotenie"))
    for meno, data in databaza.items():
        lok_bmi = bmi(data[0], data[1])
        print("{:10} {:5.2f} {}".format(meno, lok_bmi, hodnotenie(lok_bmi)))

tretiaci = {"Adela":[48,1.68], "Pavol":[92,1.89],"Peter":[59,1.84],"Zuzana":[68,1.68]}
main(tretiaci)
```

Program nech vypíše meno/á s najväčšou hmotnosťou, s najväčšiu výškou, počet záznamov s nadváhou a pod.

Príklad S.7
Napíšte program, ktorý umožní vytvoriť databázu meno : priemer. Ponuka nech umožní pridať položku do databázy, vypísať databázu, vypísať databázu utriedenú abecedne podľa mien, vypísať databázu utriedenú podľa priemerov, vypísať priemer priemerov, vypísať mená s najlepším a najhorším priemerom, odstrániť položku z databázy po zadaní mena a pod.

```
def ponuka():
    while True:
        print("Pridať žiaka ................ 1")
        print("Vypísať žiakov .............. 2")
        print("Utrieď podľa mien ........... 3")
        print("Utrieď podľa priemerov ...... 4")
        print("Priemer priemerov ........... 5")
        print("Najlepší a najhorší žiaci ... 6")
        print("Odstrániť žiaka ............. 7")
        print("Koniec ...................... ?")
        odpoved = input("Voľba: ")
        print()
        if odpoved == "1":
            pridat()
        elif odpoved == "2":
            vypisat()
        elif odpoved == "3":
            utried_mena()
        elif odpoved == "4":
            utried_priemery()
        elif odpoved == "5":
            priemer_priemerov()
        elif odpoved == "6":
```

```
            naj_ziaci()
        elif odpoved == "7":
            odstranit()
        else:
            break
        print()

def pridat():
    meno = input("Meno žiaka: ")
    priemer = float(input("Priemer žiaka: "))
    databaza[meno] = priemer
def vypisat():
    print("Databáza:")
    for (meno,priemer) in databaza.items():
        print("{:10}{:5.2f}".format(meno, priemer))

def utried_mena():
    print("Utriedené podľa mien:")
    for meno in sorted(databaza):
        print("{:10}{:5.2f}".format(meno, databaza[meno]))

def utried_priemery():
    print("Utriedené podľa priemerov:")
    zoznam = sorted(databaza.items(), key=lambda podla: podla[1])
    for prvok in zoznam:
        print("{:10}{:5.2f}".format(prvok[0], prvok[1]))

def priemer_priemerov():
    sucet = 0
    for (meno,priemer) in databaza.items():
        sucet += priemer
    if len(databaza) > 0:
        print("Priemer priemerov: {:.2f}".format(sucet/len(databaza)))

def naj_ziaci():
    najlepsi_priemer = 5.1
    najlepsi_mena = []
    najhorsi_priemer = 0.9
    najhorsi_mena = []
    for (meno,priemer) in databaza.items():
        if priemer <= najlepsi_priemer:
            if priemer < najlepsi_priemer:
                najlepsi_priemer = priemer
                najlepsi_mena = [meno]
            else:
                najlepsi_mena.append(meno)
        if priemer >= najhorsi_priemer:
            if priemer > najhorsi_priemer:
                najhorsi_priemer = priemer
                najhorsi_mena =[meno]
            else:
                najhorsi_mena.append(meno)
    print("Žiaci s najlepším priemerom:", najlepsi_mena)
    print("Žiaci s najhorším priemerom:", najhorsi_mena)

def odstranit():
    if len(databaza)>0:
        meno = input("Odstrániť žiaka: ")
        if meno in databaza:
            del databaza[meno]
        else:
            print(meno,"sa v databáze nevyskytuje!")
    else:
        print("Databáza je prázdna!")
```

```
#================================================================================
databaza = {"Zora":1.24,"Peter":1.14,"Ada":1.00,"Pavol":3.50,"Zita":1.00, "Ida":1.14}
ponuka()
```

Príklad S.8

Napíšte program, ktorý umožní vytvoriť databázu meno : známky (počty známok žiakov sú rôzne, žiak môže byť aj bez známky). Ponuka nech umožní pridať žiaka a jeho známky do databázy, vypísať mená a známky žiakov, doplniť databázu o priemery vypočítané zo známok a vypísať mená, známky a priemery žiakov a vypísať priemer priemerov.

Príklad výstupu pre voľbu 4 - Vypísať priemer priemerov:

```
Meno       Známky     Priemer
Katka      1 1          1.00
Dušan      5 4 4 2      3.75
Karol                   0.00
Fero       1 1 4        2.00
============================
Priemer                 1.69
```

Riešenie („najjednoduchší“ variant, nie je ošetrený opakovaný výber ľubovoľnej voľby!):

```
def ponuka():
    print()
    print("PROGRAM NIE JE OŠETRENÝ NA OPAKOVANÝ VÝBER ĽUBOVOĽNEJ VOĽBY!!!")
    print()
    while True:
        print("Pridať žiaka alebo zmeniť známky ..... 1")
        print("Vypísať mená a známky ..... 2")
        print("Vypísať mená, známky a priemery ..... 3")
        print("Vypísať priemer priemerov (najprv 3) ..... 4")
        print("Koniec ..... ?")
        odpoved = input("Voľba: ")
        print()
        if odpoved == "1":
            pridatZiaka()
        elif odpoved == "2":
            vypisatMenaZnamky()
        elif odpoved == "3":
            vypocitajAdoplnPriemery()
            vypisatMenaZnamkyPriemery()
        elif odpoved == "4":
            vypisatPriemerPriemerov()
        else:
            break
        print()

def pridatZiaka():
    meno = input("Meno žiaka: ")
    if meno in kl_harok.keys():
       print("Upozornenie: Meno {} sa už v zozname vyskytuje, zmení mu známky!".format(meno))
    znamky = []
    while True:
        znamkaStr = input("Známka alebo Enter: ")
        if len(znamkaStr) > 0:
            znamky.append(int(znamkaStr))
        else:
            break
    kl_harok[meno] = znamky

def vypisatMenaZnamky():
    for meno, znamky in kl_harok.items():
        print("{:10}".format(meno), end="")
        for i in range(len(znamky)):
            print("{:2}".format(znamky[i]), end="")
```

```
        print()

def vypocitajAdoplnPriemery():
    for meno, znamky in kl_harok.items():
        sucet = 0
        for znamka in znamky:
            sucet += int(znamka)
        if len(znamky) > 0:
            kl_harok[meno].append(sucet/len(znamky))
        else:
            kl_harok[meno].append(0.00)

def vypisatMenaZnamkyPriemery():
    print("{:10} {:10} {:8}".format("Meno","Známky","Priemer"))
    for meno, znamkyApriemer in kl_harok.items():
        znamkyStr = ""
        for i in range(len(znamkyApriemer)-1):
            znamkyStr += str(znamkyApriemer[i]) + " "
        print("{:10} {:10} {:6.2f}".format(meno, znamkyStr, znamkyApriemer[-1]))

def vratPriemerPriemerov():
    sucet = 0
    for meno, znamkyApriemer in kl_harok.items():
        sucet += znamkyApriemer[-1]
    if len(kl_harok) > 0:
        return sucet/len(kl_harok)
    else:
        return 0.0

def vypisatPriemerPriemerov():
    vypisatMenaZnamkyPriemery()
    print(28*"=")
    print("{} {:20.2f}".format("Priemer", vratPriemerPriemerov()))

# ========================================================================
kl_harok = {"Fero":[1,1,4],"Katka":[1,1],"Karol":[],"Dušan":[5,4,4,2]}
ponuka()
```

Program príkladu S.8 doplňte o výpis mien, ktoré nemajú ani jednu známku, o výpis mien s najlepším a najhorším priemerom, o ponuku odstrániť položku z databázy po zadaní mena a pod.

Príklad S.9
Vytvorte program umožňujúci zakódovať zadaný text Morseho kódom, zakódovaný text - správu „odoslať“ (prehrať zvukovo).

```
import winsound

"""
písmená sa oddeľujú /, slová //
zvuk: winsound.Beep(frekvencia, dĺžka trvania tónu)
frekvencia: 37–32767Hz, dĺžka trvania tónu: v milisekundách
"""

morse = {'a':'.-','b':'-...','c':'-.-.','d':'-..','e':'.','f':'..-.','g':'--
.','h':'....','ch':'----','i':'..','j':'.---','k':'-.-','l':'.-..','m':'--','n':'-
.','o':'---','p':'.--.','q':'--.-','r':'.-.','s':'...','t':'-','u':'..-','v':'...-
','w':'.--','x':'-..-','y':'-.--','z':'--..','1':'.----','2':'..---','3':'...--
','4':'....-','5':'.....','6':'-....','7':'.--...','8':'---..','9':'----.','0':'-----
',',':'.-.-.-',',':'--..--','?':'..--..','!':'--..-'}


def zakoduj(text):
    result = ""
```

```
    for znak in text:
        if znak in morse:
            result += morse[znak] + "/"
        elif znak == " ":
            result += "/"
        else:
            result += "?/"
    return result

def odosli(sprava):
    for znak in sprava:
        if znak == ".":
            winsound.Beep(880,150)
        elif znak == "-":
            winsound.Beep(880,500)
        else:
            winsound.Beep(37,500)

# ============================================================

text_na_zakodovanie = input("Text na zakódovanie: ").lower()
zakodovany_text = zakoduj(text_na_zakodovanie)
print(zakodovany_text)
odosli(zakodovany_text)
```

Príklad doplňte o funkciu `dekoduj(zakodovany_text)`, ktorá prekóduje zakódovaný text na znaky.
Vety sa oddeľujú v Morseho kóde /// - doplňte program o túto vlastnosť.
**Príklad doplňte tak, aby nebolo možné zadať z klávesnice iný znak ako kľúč slovníka morse.

Príklad S.10
Vytvorte program, ktorý zašifruje zadaný text podľa vygenerovanej tabuľky (postupnosť bežne používaných znakov zamieša a použije ako šifru - prvý znak z postupnosti znakov bude šifrovať prvým znakom zo šifry atď.). Ak sa šifrovaný znak v tabuľke nenachádza, nahradí ho mriežkou (#).
Vytvorte aj dešifrovaciu tabuľku a funkciu na odšifrovanie zašifrovanej správy. Pri dešifrovaní mriežku dešifruje ako mriežku.
Príklad výstupu:

```
Zašifrovať: Kontakt dnes o 17:30, miesto C.
Zašifrovaná správa:
ťuaoňťof8ařsfufx7k!zlfygŕsoufšý          # napr. medzera je nahradená f
Dešifrovaná správa:
kontakt dnes o 17:30, miesto c.
import random

def vytvorSifrovaciuTabulku(znaky):
    sifra = list(znaky)
    random.shuffle(sifra)                   # náhodne sa premiešajú dovezené znaky
    sif_tabulka = {}
    for i in range(len(znaky)):
        sif_tabulka[znaky[i]] = sifra[i]    # vytvorí sa šifrovacia tabuľka
    return sif_tabulka

def zasifruj(text, tabulka):
    text = text.lower()
    zasifrovany = ""
    for znak in text:
        if znak in tabulka:
            zasifrovany += tabulka[znak]
        else:
            zasifrovany += "#"
    return zasifrovany

def vytvorDesifrovaciuTabulku(tabulka):
```

```
    desif_tabulka = {}
    for kluc, hodnota in tabulka.items():
        desif_tabulka[hodnota] = kluc
    return desif_tabulka

def desifruj(text, tabulka):
    desifrovany = ""
    for znak in text:
        desifrovany += tabulka[znak]
    return desifrovany

# ================================================================================
pouzite_znaky = "aáäbcčdďeéfghiíjklľĺmnňoóôpqrŕsštťuúvwxyýzž .,;:?!0123456789"
s_tabulka = vytvorSifrovaciuTabulku(pouzite_znaky)
sprava = input("Zašifrovať: ")
zasifrovana_sprava = zasifruj(sprava, s_tabulka)
print("Zašifrovaná správa:")
print(zasifrovana_sprava)

d_tabulka = vytvorDesifrovaciuTabulku(s_tabulka)
desifrovana_sprava = desifruj(zasifrovana_sprava, d_tabulka)
print("Dešifrovaná správa:")
print(desifrovana_sprava)
```

Príklad S.11

Vytvorte program (tzv. mincovku, pozri program T.1 tohto študijného textu!), ktorý po zadaní databázy meno zamestnanca a jeho platu (celé číslo) vypíše minimálny počet euro bankoviek a mincí potrebných na vyplatenie zadanej sumy a na záver vypíše aj celkový sumár potrebných bankoviek a mincí pre všetkých zamestnancov.

Ukážka výpisu:

```
Darina   1071 €
 1000 € ....  1
   50 € ....  1
   20 € ....  1
    1 € ....  1
===============
Adam      999 €
  500 € ....  1
  200 € ....  2
   50 € ....  1
   20 € ....  2
    5 € ....  1
    2 € ....  2
===============
Peter    1234 €
 1000 € ....  1
  200 € ....  1
   20 € ....  1
   10 € ....  1
    2 € ....  2
===============
MINCOVKA-SUMÁR:
 5000 € ....  0
 1000 € ....  2
  500 € ....  1
  200 € ....  3
  100 € ....  0
   50 € ....  2
   20 € ....  4
   10 € ....  1
    5 € ....  1
    2 € ....  4
    1 € ....  1
```

Riešenie:

```
HODNOTY_MENY = (5000,1000,500,200,100,50,20,10,5,2,1)
celkovyPocetHodnoty = {hodnota:0 for hodnota in HODNOTY_MENY}

def vratPocetHodnota(suma,hodnota):
    return suma//hodnota

def vratNovuSumu(suma, hodnota):
    return suma%hodnota

def mincovka_jeden(suma):
    for hodnota in HODNOTY_MENY:
        pocet = vratPocetHodnota(suma, hodnota)
        if pocet > 0:
            celkovyPocetHodnoty[hodnota] += pocet
            print("{:5d} € .... {:2d}".format(hodnota, pocet))
        suma = vratNovuSumu(suma, hodnota)

def mincovka_vsetci():
    for meno in zamestnanci:
        print("{:7s} {:5d} €".format(meno,zamestnanci[meno]))
        mincovka_jeden(zamestnanci[meno])
        print(15*"=")

    print("MINCOVKA-SUMÁR:")
    for hodnota in HODNOTY_MENY:
        print("{:5d} € .... {:2d}".format(hodnota, celkovyPocetHodnoty[hodnota]))

# ================= HLAVNÝ PROGRAM ==================
zamestnanci = {'Peter':1234, 'Adam':999, 'Darina':1071}
mincovka_vsetci()
```