Dvojrozmerné pole

Názov tohto údajového typu si opäť „požičiame“ z iných programovacích jazykov, v ktorých, na základe konštrukcie, sa nazýva aj pole polí. V Pythone to je zoznam zoznamov („listy v liste“:) alebo dvojrozmerný zoznam. Konštrukcia dvojrozmerného poľa je nasledujúca: zoberieme jednorozmerné pole (údajový typ list - zoznam) a každý prvok tohto jednorozmerného poľa bude opäť nejaké jednorozmerné pole. V Pythone asi niečo takéhoto [[vnútorný_zoznam na indexe 0], [vnútorný_zoznam na indexe 1], [vnútorný_zoznam na indexe 2], ...], pričom každý vnútorný_zoznam na nejakom indexe má svoje prvky s indexmi 0, 1,... To vedie k dvojrozmernej tabuľke s prvkami tab[i][j] (prvok v riadku s indexom i a v stĺpci s indexom j) pre všetky dovolené hodnoty i a j.

	1.stĺ. (ind.0)	**2.stĺpec**	**3.stĺpec**	**...**	**stĺpec s ind.j**		
1.riadok (ind.0)	tab[0][0]	tab[0][1]	tab[0][2]	...	tab[0][j]	...	tab[0][M_0]
2.riadok (ind.1)	tab[1][0]	tab[1][1]	tab[1][2]	...	tab[1][j]	...	tab[1][M_1]
3.riadok (ind.2)	tab[2][0]	tab[2][1]	tab[2][2]	...	tab[2][j]	...	tab[2][M_2]
...	...	...	...	...	...	...	...
riadok s ind.i	tab[i][0]	tab[i][1]	tab[i][2]	...	**tab[i][j]**	...	tab[i][M_i]
...	...	...	...	...	...	...	...
riadok s ind.N-1	tab[N-1][0]	tab[N-1][1]	tab[N-1][2]	...	tab[N-1][j]	...	tab[N-1][M_{N-1}]

pričom počet stĺpov v jednotlivých riadkoch síce nemusí byť rovnaký, ale najčastejšie býva rovnaký, t.j. $M_0 = M_1 = M_2 = ... = M_{N-1}$. Matematici takúto tabuľku s N riadkami a M stĺpcami nazývajú **matica** typu N x M a jej prvky, pre maticu pomenovanú písmenom **a,** označujú $a_{i,j}$, napríklad $a_{0,2}$. Ak sa počet riadkov rovná počtu stĺpcov, hovoria o štvorcovej matici.

Príklad vytvorenia dvojrozmerného zoznamu

```
>>> zoz0 = [1,2,3]
>>> zoz1 = ['A','Z']
>>> zoz = [zoz0, zoz1]
>>> zoz
[[1, 2, 3], ['A', 'Z']]
>>> zoz[0][0]
1
>>> zoz[0][1]
2
>>> zoz[1][0]
'A'
>>> zoz[1][1]
'Z'
>>> zoz[2][1]
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    zoz[2][1]
IndexError: list index out of range
```

Iný zápis

```
>>> dvojpole = [[1],[2,2],[3,3,3],[4,4,4,4],[5,5,5,5,5]]
>>> dvojpole
[[1], [2, 2], [3, 3, 3], [4, 4, 4, 4], [5, 5, 5, 5, 5]]
>>> type(dvojpole)
<class 'list'>
```

Úloha D.1.1

Vytvorte funkciu, ktorá dvojrozmerné pole vypíše vo forme tabuľky.

Riešenie

Vypísanie dvojrozmerného poľa vo forme tabuľky znamená nastaviť index prvého riadka (i = 0) a prejsť celý riadok (j = 0, 1, 2, ..., počet prvkov v riadku), pričom sa vypisujú hodnoty poľa v danom riadku; následne nastaviť index druhého riadka a opäť prejsť celý druhý riadok (j = 0, 1, 2, ..., počet prvkov v riadku) atď., až rovnakú operáciu spraviť s posledným poľom - riadkom dvojrozmerného poľa. Nevyhnutné je použiť dva cykly, prvý cyklus nastaví a postupne mení riadok, ktorý sa vypisuje, a druhý cyklus nastaví a mení indexy stĺpcov vo vypisovanom riadku od prvého po posledný v danom riadku. Cyklus `for i in range(len(tab)):` nastaví index riadka na 0, potom

na 1 atď. až na index posledného riadka. Cyklus `for j in range(len(tab[i])):` nastaví index stĺpca na 0, potom na 1 atď. až na index posledného stĺpca v i-tom riadku. Keďže pre každý riadok treba prejsť všetky stĺpce v riadku, druhý cyklus je vložený do prvého. Takéto usporiadanie cyklov nazývame cyklus v cykle.

```
def vypisDvojPole(tab):
    for i in range(len(tab)):                          # nastavuje index riadku
        for j in range(len(tab[i])):                   # nastavuje index stĺpca v i-tom riadku
            print ("{:3}".format(tab[i][j]), end=" ")  # zobrazí hodnotu prvku tab[i][j]
        print()                                        # nastavenie vypisovania na nový riadok
```

Príkazy

```
def main():
    tab = [[1],[2,2],[3,3,3],[4,4,4,4],[5,5,5,5,5]]
    vypisDvojPole(tab)

main()
```

nám dajú požadovaný výpis

```
  1
  2   2
  3   3   3
  4   4   4   4
  5   5   5   5   5
```

Ostatné riešenie predstavuje klasický (z hľadiska programovacích jazykov univerzálnejší) „indexový“ prístup. Python umožňuje aj bezindexové riešenie, ktorého autorom je študent Patrik Šimanský:

```
for vnut_zoznam in zoznam:                 # postupne ber všetky vnútorné zoznamy - riadky tabuľky
    for prvok in vnut_zoznam:              # postupne ber prvky nastaveného vnútorného zoznamu
        print(prvok, end=" ")              # vypíš hodnotu a medzeru (zrušený výpis na nový riadok)
    print()                                # pre nový vnut_zoznam nastavenie výpisu na nový riadok
```

Úloha D.1.2.a

Napíšte funkciu, ktorá vytvorí dvojrozmerné pole (maticu) obsahujúce náhodné celé čísla zo zadaného intervalu. Parametrami funkcie nech je počet riadkov a počet stĺpcov poľa a dolná a horná hranica intervalu, z ktorého má generovať celé čísla.

Riešenie

Najprv použijeme „klasický konštrukčný“ prístup.

```
import random

def vytvorDvojPole(r=10, s=10, dh=0, hh=99):
    tab = []                    # vytvorenie prázdneho vonkajšieho zoznamu
    for i in range(r):
        tab.append([])          # pridanie prázdneho vnútorného zoznamu ako nového prvku do vonk. zoznamu
        for j in range(s):
            tab[i].append(random.randint(dh, hh))      # pridanie hodnoty do vnútorného zoznamu
    return tab
```

Príkazy

```
def main():
    tab = vytvorDvojPole(4,7,0,99)
    vypisDvojPole(tab)

main()
```

vytvoria a vypíšu dvojrozmerné pole napríklad s hodnotami

```
 63  28   9  91  15  88  53
 22  89  96  26  44   0   3
 84   5  72  28  54  82  28
 58  21  12  86  38  33   6
```

O správnosti našich úvah sa môžeme presvedčiť aj zavolaním upravenej funkcie `vytvorDvojPole(...)`

```
def vytvorDvojPole(r=10, s=10, dh=0, hh=99):
    tab = []
    for i in range(r):
        tab.append([])
        for j in range(s):
            pass                    # prázdny príkaz
    return tab
```

kde pri použití príkazov

```
def main():
    tab = vytvorDvojPole(4,7,-9,9)
    print(tab)
```

dostávame „konštrukčný“ výpis

```
[[], [], [], []]
```

Iná z možností vytvorenia dvojrozmerného poľa (autor Patrik Šimanský)

```
def vytvorDvojPoleSim(r=10,s=10,dh=0,hh=99):
    tab = []
    for i in range(r):
        vnut_zoznam = []                    # vytvorenie vnútorného zoznamu (nasledujúce tri riadky)
        for j in range(s):
            vnut_zoznam.append(random.randint(dh, hh))
        tab.append(vnut_zoznam)             # pridanie vytvoreného vnútorného zoznamu do zoznamu
    return tab
```

Najkratšie riešenie funkcie vytvorDvojPole(r=10, s=10, dh=0, hh=99) (rozlúšti zápis sprava doľava!:):

```
    return [[random.randint(dh,hh) for j in range(s)]2 for i in range(r)]1
```

[1] cyklus for i vytvorí zoznam obsahujúci r („er“ - počet riadkov) cyklom for j postupne vytvorených zoznamov
[2] cyklus for j vytvorí zoznam obsahujúci s („es“ - počet stĺpcov) náhodných celých čísel z intervalu <dh,hh>

Úloha D.1.2.b

Napíšte funkciu, ktorá vytvorí dvojrozmerné pole obsahujúce náhodné celé čísla zo zadaného intervalu. Nech počet prvkov v každom riadku je náhodný - generovaný z intervalu <1,s>. Parametrami funkcie nech je počet riadkov a maximálny počet stĺpcov poľa a dolná a horná hranica intervalu, z ktorého má generovať celé čísla.

Možný výstup napríklad pre r = 4, s = 7, dh = -9 a hh = 9

```
 -3 -8  5 -9  3                          # vygenerovaných päť prvkov
 -4                                      # vygenerovaný jeden prvok
 -3  4 -3  1 -7  0                       # vygenerovaných šesť prvkov
  9  4  2                                # vygenerované tri prvky
```

Riešenie

```
def vytvorMixDvojPole(r,s,dh,hh):
    return [[random.randint(dh,hh) for j in range(random.randint(1,s))] for i in range(r)]
```

Použitie napríklad `tab = vytvorMixDvojPole(4,7,-9,9).`

Úloha

Vytvorte funkciu, ktorá vygeneruje rovnaké trojuholníkové dvojrozmerné pole, aké sme použili v úlohe D.1.1. Parametrom funkcie nech je aj s - „šírka základne trojuholníka“, t.j. napríklad pre s = 8 bude posledný riadok výpisu obsahovať hodnoty 8 8 8 8 8 8 8 8.

V ďalších úlohách sa už budeme zaoberať len dvojrozmernými poľami (dvojrozmernými zoznamami), v ktorých je v každom riadku rovnaký počet prvkov (stĺpcov), matematicky povedané maticami a budeme predpokladať, že pole obsahuje aspoň jednu hodnotu.

Ukážme si, aké zadania dvojrozmerného poľa budeme používať:
1. dvojrozmerné pole zadané vymenovaním prvkov,
2. vygenerované dvojrozmerné pole,
3. dvojrozmerné pole načítané z textového súboru.

Dvojrozmerné pole zadané vymenovaním prvkov má napríklad tvar

```
tab = [ [0,15,27,0,0],          # hodnoty poľa v prvom riadku
        [15,0,34,2,25],         # hodnoty v druhom riadku atď.
        [27,34,0,9,12],
        [0,2,9,0,76],
        [0,25,12,76,0] ]
```

v tomto prípade to je štvorcová matica typu 5 x 5. Zápis do viacerých riadkov nie je nevyhnutný, je však prehľadnejší.

Dvojrozmerné pole vytvorené generovaním celočíselných hodnôt sme si ukázali v úlohách D.1.2 so záverom, že vystačíme so zápisom (prípadne jeho modifikáciou)

```
[[random.randint(dh,hh) for j in range(s)] for i in range(r)]
```

kde počet riadkov r, počet stĺpcov s a interval <dh,hh>, z ktorého sú náhodne vyberané hodnoty, sú zadané alebo počet stĺpcov s sa generuje pre každý riadok zvlášť.
Zrejme vygenerovať by sme dokázali ešte dvojrozmerné pole reálnych čísel prípadne znakov, ale so zmysluplnými hodnotami typu str by vo všeobecnosti boli problémy (napríklad keby dvojrozmerné pole malo obsahovať mená). V týchto situáciách použijeme dvojrozmerné pole zadané vymenovaním prvkov alebo načítaním zo súboru.

Dvojrozmerné pole načítané z textového súboru
Ak ste pochopili „konštrukčné“ riešenie z úlohy D.1.2.a a ovládate prácu so súborom, pochopenie nasledujúcej funkcie vám nespôsobí problémy ☺

```
def nacitajZoSuboruDoDvojPola(nazov_suboru):
    with open(nazov_suboru,"r") as f:
        tab = []
        r = 0
        for riadok in f:
            hodnotyStr = riadok.split()
            tab.append([])
            for s in range(len(hodnotyStr)):
                tab[r].append(int(hodnotyStr[s]))
            r += 1
    return tab
```

Príklad použitia (nezabudnite najprv vytvoriť súbor tabulka.txt)

```
def main():
    m = nacitajZoSuboruDoDvojPola("tabulka.txt")
    vypisDvojPole(m)

main()
```

Elegantné pythonovské „jednoriadkové“ riešenie:

```
def nacitajZoSuboruDoDvojPola(nazov_suboru):
    with open(nazov_suboru,"r") as f:
        return [[int(riadok.split()[s]) for s in range(len(riadok.split()))] for riadok in f]
```

Odteraz začneme používať modul dvojpole, ktorý obsahuje funkcie vytvorDvojPole() a vypisDvojPole().

```
>>> import dvojpole
>>> help(dvojpole)
Help on module dvojpole:

NAME
    dvojpole - # modul dvojpole
```

```
FUNCTIONS
    vypisDvojPole(tab)
        Vypíše dvojrozmerné pole vo forme tabuľky

    vytvorDvojPole(r=10, s=10, dh=0, hh=99)
        Vytvorí dvojrozmerné pole r x s s náhodnými celočíselnými hodnotami z <dh,hh>
```

Úloha D.2.1
Vytvorte funkciu, ktorá vráti aritmetický priemer hodnôt dvojrozmerného číselného poľa.

Riešenie

```
import dvojpole

def vratPriemer(tab):
    sucet, pocet = 0, 0
    for riadok in tab:
        sucet += sum(riadok)
        pocet += len(riadok)
    return sucet/pocet

def vratPriemerKlas(tab):                    # klasické riešenie cez indexy
    sucet = 0
    pocet = 0
    for i in range(len(tab)):
        for j in range(len(tab[i])):
            sucet += tab[i][j]
            pocet += 1
    return sucet/pocet

def main():
    m = dvojpole.vytvorDvojPole(2,3)
    dvojpole.vypisDvojPole(m)
    print("Priemer všetkých prvkov: {:.2f}".format(vratPriemer(m)))
    print("Priemer všetkých prvkov: {:.2f}".format(vratPriemerKlas(m)))

main()
```

Úloha D.2.2
Vytvorte funkciu, ktorá vráti aritmetické priemery hodnôt v jednotlivých riadkoch dvojrozmerného číselného poľa.

Riešenie

```
import dvojpole

def vratPriemeryVriadkoch(tab):
    priemery = []
    for riadok in tab:
        priemery.append(sum(riadok)/len(riadok))
    return priemery

def main():
    m = dvojpole.vytvorDvojPole(2,3)
    dvojpole.vypisDvojPole(m)
    print(vratPriemeryVriadkoch(m))

main()

def vratPriemeryVriadkochKlas(tab):          # klasické riešenie cez indexy
    priemery = []
    for i in range(len(tab)):
        sucet = 0
        pocet = len(tab[i])
        for j in range(pocet):
            sucet += tab[i][j]
```

```
        priemery.append(sucet/pocet)
    return priemery
```

Úloha D.2.3
Vytvorte funkciu, ktorá vráti aritmetické priemery hodnôt v jednotlivých stĺpcoch dvojrozmerného číselného poľa. Počet prvkov v každom riadku je rovnaký!

Riešenie

```
import dvojpole

def vratPriemeryVstlpcoch(tab):
    priemery = []
    for j in range(len(tab[0])):
        sucet = 0
        for i in range(len(tab)):
            sucet += tab[i][j]
        priemery.append(sucet/len(tab))
    return priemery

def main():
    m = dvojpole.vytvorDvojPole(2,3)
    dvojpole.vypisDvojPole(m)
    print(vratPriemeryVstlpcoch(m))

main()
```

Úloha D.3.1
Vytvorte funkciu, ktorá vráti najväčší prvok dvojrozmerného poľa.

Riešenie

```
import dvojpole

def vratMax(tab):
    tabmax = tab[0][0]              # použitie názvu max je pre interpreter mätúce
    for riadok in tab:
        rmax = max(riadok)
        if rmax > tabmax:
              tabmax = rmax
    return tabmax

def main():
    m = dvojpole.vytvorDvojPole(4,7,0,99)
    dvojpole.vypisDvojPole(m)
    print("Najväčší prvok", vratMax(m))

main()
```

Úloha D.3.2
Vytvorte funkciu, ktorá vráti najväčší prvok dvojrozmerného poľa a počet jeho výskytov v poli.

Riešenie

```
def vratMaxApocet(tab):
    tabmax = tab[0][0]
    pocet = 0
    for riadok in tab:
        rmax = max(riadok)
        if rmax > tabmax:
            tabmax = rmax
            pocet = riadok.count(tabmax)
        else:
            if rmax == tabmax:
                pocet += riadok.count(tabmax)
    return tabmax, pocet
```

```
    '''
    # „klasické“ riešenie bez použitia funkcie count()
    for i in range(len(tab)):
        for j in range(len(tab[i])):
            if tab[i][j] >= tabmax:
                if tab[i][j] > tabmax:
                    tabmax = tab[i][j]
                    pocet = 1
                else:
                    pocet += 1
    return tabmax, pocet
    '''

def main():
    m = dvojpole.vytvorDvojPole(4,7,0,99)
    dvojpole.vypisDvojPole(m)
    max, pocet = vratMaxApocet(m)
    print("Najväčší prvok je {} a vyskytuje sa {}-krát.".format(max, pocet))
```

Úloha D.3.3
Vytvorte funkciu, ktorá vráti najväčší prvok dvojrozmerného poľa a indexy jeho výskytov v poli.

Riešenie

```
def vratMaxAindexy(tab):
    tabmax = tab[0][0]
    maxindexy = [[0,0]] #zoz.index(x)
    for i in range(len(tab)):
        for j in range(len(tab[i])):
            if tab[i][j] >= max:
                if tab[i][j] > max:
                    max = tab[i][j]
                    pocet = 1
                else:
                    pocet += 1
    return max, pocet
```

Úloha D.3.4
Vytvorte funkciu, ktorá vráti najväčší prvok každého riadku dvojrozmerného poľa.

Riešenie

```
def vratMaxVriadkoch(tab):
    maxima = []
    for i in range(len(tab)):
        maxima.append(max(tab[i]))
    return maxima
```

Úloha D.3.5
Vytvorte funkciu, ktorá vráti najväčší prvok každého stĺpca dvojrozmerného poľa.

Riešenie

```
def vratMaxVstlpcoch(tab):
    '''matica'''
    maxima = []
    for j in range(len(tab[0])):
        smax = tab[0][j]
        for i in range(1,len(tab)):
            if tab[i][j] > smax:
                smax = tab[i][j]
        maxima.append(smax)
    return maxima
```

Vytvorenie kópie dvojrozmerného poľa

Často potrebujeme, aby nám zostalo k dispozícii aj pôvodné pole.

```
kopia = tab[:] # nie je riešením!

def vytvorKopiu(tab):
    '''
    # pre pochopenie:)
    kopia = []
    for i in range(len(tab)):
        kopia.append([])
        for j in range(len(tab[i])):
            kopia[i].append(tab[i][j])
    return kopia
    '''
    # skrátený zápis
    return [tab[i][:] for i in range(len(tab))]
```

Úloha D.4.1

Vytvorte funkciu, ktorá utriedi vzostupne prvky v jednotlivých riadkoch dvojrozmerného poľa.
Príklad výstupu:

```
Pôvodná:
 13  12  43  27  59  62  83
  5  79  66  93  72  87  84
 95  26  33  84  25  97  27
 90  38   5  45  38  24   4
Utriedená:
 12  13  27  43  59  62  83
  5  66  72  79  84  87  93
 25  26  27  33  84  95  97
  4   5  24  38  38  45  90
```

Riešenie

```
import dvojpole

def utriedVriadkoch(tab):
    for i in range(len(tab)):
        tab[i].sort()

def vytvorKopiu(tab):
    return [tab[i][:] for i in range(len(tab))]

def main():
    m = dvojpole.vytvorDvojPole(4,7,0,9)
    dvojpole.vypisDvojPole(m)
    kopia = vytvorKopiu(m)
    utriedVriadkoch(kopia)
    print("Matica utriedená v riadkoch")
    dvojpole.vypisDvojPole(kopia)
    print("Pôvodná matica")
    dvojpole.vypisDvojPole(m)

main()
```

Ďalšie možné riešenia (bez použitia štandardnej funkcie sort):

```
def utriedRiadok(riadok):
    N = len(riadok)
    for pp in range(1,N):                  # bubblesort
        for i in range(N-pp):
                if riadok[i]>riadok[i+1]:
                    riadok[i],riadok[i+1] = riadok[i+1],riadok[i]
```

```
def utriedVriadkoch(tab):
    for riadok in tab:
        utriedRiadok(riadok)
```

Použitie

```
m = dvojpole.vytvorDvojPole(4,7,0,99)
k = dvojpole.vytvorKopiu(m)
print("Pôvodná:")
dvojpole.vypisDvojPole(m)
utriedVriadkoch(k)
print("Utriedená:")
dvojpole.vypisDvojPole(k)
```

alebo „klasické“ riešenie:

```
def utriedVriadkochKlas(tab):
    M = len(tab[0])                         # (dohovor) v každom riadku je rovnaký počet stĺpcov!
    for i in range(len(tab)):
        for pp in range(1,M):               # bubblesort
            for j in range(M-pp):
                if tab[i][j]>tab[i][j+1]:
                    tab[i][j],tab[i][j+1] = tab[i][j+1],tab[i][j]
```

Úloha D.4.2

Vytvorte funkcia, ktorá utriedi riadky dvojrozmerného poľa podľa hodnôt v určenom stĺpci.

Príklad výstupu:

```
Utriediť podľa stĺpca: 4
Riadky utriedené podľa hodnôt v 4. stĺpci
  1   2   7   2   5   4   5
  7   3   9   7   8   8   1
  5   2   6   8   1   2   5
  5   1   6   9   4   4   8
Pôvodná matica
  5   1   6   9   4   4   8
  5   2   6   8   1   2   5
  7   3   9   7   8   8   1
  1   2   7   2   5   4   5
```

Riešenie

```
def utriedRiadkyPodlaStlpca(tab, stl=0):
    for pp in range(1,len(tab)):                # bubblesort
        for i in range(1,len(tab)):             # porovnávame "i-1 s i"!
            if tab[i-1][stl] > tab[i][stl]:
                    tab[i-1], tab[i] = tab[i], tab[i-1]

def main():
    m = dvojpole.vytvorDvojPole(4,7,0,9)
    dvojpole.vypisDvojPole(m)
    kopia  = vytvorKopiu(m)
    podla = 0
    utriedRiadkyPodlaStlpca(kopia, podla)
    print("Riadky utriedené podľa hodnôt v {}. stĺpci".format(podla+1))
    dvojpole.vypisDvojPole(kopia)
```

Úloha D.4.3

Vytvorte funkciu, ktorá utriedi vzostupne prvky v jednotlivých stĺpcoch dvojrozmerného poľa.

Riešenie

```
import dvojpole

def utriedVstlpcoch(tab):
    pocetStlpcov = len(tab[0])
    for j in range(pocetStlpcov):
        for pp in range(pocetStlpcov-1):
            for i in range(1,len(tab)-pp):
                if tab[i-1][j] > tab[i][j]:
                    tab[i-1][j], tab[i][j] = tab[i][j], tab[i-1][j]

def main():
    m = dvojpole.vytvorDvojPole(4,7,0,9)
    dvojpole.vypisDvojPole(m)

    kopia  = vytvorKopiu(m)
    utriedVstlpcoch(kopia)
    print("Matica utriedená v stĺpcoch")
    dvojpole.vypisDvojPole(kopia)

main()
```

Možno vám napadlo, že **vytvorením matice, v ktorej by riadky obsahovali hodnoty zo sĺpcov pôvodnej matice** („otočenie“ matice o 90°), by umožnilo použiť funkcie, ktoré hľadajú vlastnosti v riadkoch, aj pre hľadanie tých istých vlastností v stĺpcoch pôvodnej matice. Tým by sme ušetrili programovanie napríklad úloh D.3.5 a D.4.3.

Úloha D.5.1
Vytvorte funkcia, ktorá hodnoty zo stĺpcov matice uloží do riadkov, t.j. tab[i][j] sa bude rovnať tab[j][i] v novej matici (z matice typu NxM vznikne „otočená“ matica typu MxN).
Príklad výstupu:

```
Pôvodná mativca:
  6   3   5   8   2   1   4
  9   1   8   9   5   5   2
  9   3   8   0   1   9   4
  1   7   2   0   8   3   8
"Otočená" matica:
  6   9   9   1
  3   1   3   7
  5   8   8   2
  8   9   0   0
  2   5   1   8
  1   5   9   3
  4   2   4   8
```

Riešenie

```
def otocMaticu(tab):
    r = len(tab)
    s = len(tab[0])
    otab = []
    for j in range(s):
        otab.append([])
        for i in range(r):
            otab[j].append(tab[i][j])
    return otab

def main():
    mat = dvojpole.vytvorDvojPole(4,7,0,9)
    mat1 = dvojpole.vytvorKopiu(mat)
    mat1 = otocMaticu(mat1)
    print("Pôvodná matica:")
    dvojpole.vypisDvojPole(mat)
    print('"Otočená" matica:')
    dvojpole.vypisDvojPole(mat1)
```

```
main()
```

Hypotézu vyslovenú pred úlohou D.5.1 overte v praxi.

Úloha D.6.1
V mape sú uvedené nadmorské výšky krajiny namerané v bodoch zvolenej pravouhlej siete (hodnoty od -10 po 500 m n.m.). Vytvorte a použite funkciu, ktorá vypíše priemernú nadmorskú výšku krajiny.

```
mapa = [[3,10,15,27,19,54,95,160,146,263],
        [96,65,175,253,177,367,245,187,199,165],
        [-10,-2,0,23,187,254,376,402,499,378],
        [-8,-5,-1,4,15,15,15,15,87,123],
        [0,0,10,28,48,167,222,109,75,39],
        [0,1,5,27,19,54,95,160,46,63],
        [96,65,175,253,177,367,245,187,199,165],
        [-10,-2,0,23,187,254,376,402,492,378],
        [-8,-5,-1,4,15,15,15,13,87,123],
        [0,-3,10,28,48,167,222,109,75,37]]
```

Opäť uvádzame klasické aj „pythonovské“ riešenie:

```
def vratPriemNadVyskuKlas(mapa):
    N = len(mapa)
    M = len(mapa[0])
    sucet = 0;
    for i in range(N):
        for j in range(M):
            sucet += mapa[i][j]
    return sucet/N/M

def vratPriemNadVysku(mapa):
    sucet = 0
    for riadok in mapa:
        sucet += sum(riadok)
    return sucet/len(mapa)/len(mapa[0])

print("Priemerná výška: {:.2f} m n.m.".format(vratPriemNadVyskuKlas(mapa)))
print("Priemerná výška: {:.2f} m n.m.".format(vratPriemNadVysku(mapa)))
```

Výstup:
```
Priemerná výška: 112.36 m n.m.
Priemerná výška: 112.36 m n.m.
```

Úloha D.6.2
V mape z úlohy D.6.1 sú uvedené nadmorské výšky krajiny namerané v bodoch zvolenej pravouhlej siete (hodnoty od -10 po 500 m n.m.). Vytvorte a použite funkciu, ktorá vypíše, koľko % krajiny tvorí súš, t.j. nadmorská výška je väčšia alebo rovná nule.

```
def vratPercentaSuse(graf):
    N = len(graf)
    sus = 0
    for i in range(N):
        for j in range(N):
            if graf[i][j] >= 0: sus +=1
    return sus/N/N*100

print("Súš zaberá {:.2f} %".format(vratPercentaSuse(mapa)))
```

Úloha D.6.3
V mape z úlohy D.6.1 sú uvedené nadmorské výšky krajiny namerané v bodoch zvolenej pravouhlej siete (hodnoty od -10 po 500 m n.m.). Vytvorte a použite funkciu, ktorá vypíše prevýšenia v smere zo severozápadu na juhovýchod (sever je „hore“). Prevýšenie sa počíta ako rozdiel dvoch susedných nadmorských výšok v sledovanom smere, napríklad medzi bodmi 3 a 65 je prevýšenie 65 - 3 = 62 (stúpanie), medzi bodmi 65 a 0 je prevýšenie 0 - 65 = -65 (klesanie).