**Dvojrozměrná tabulka**

Je seznam kde každý jeho prvek je seznamem. Tyto seznamy vytvářejí dvojrozměrnou tabulku, kde jednotlivé pole prvního seznamu vytvářejí sloupce a jednotlivá pole vnořených seznamů vytvářejí řádky (dle indexu). Je možné vytvářet i troj a více rozměrné tabulky, kde u trojrozměrné tabulky se již dostáváme do 'pixelizace' 3D prostoru.

Dvojrozměrné údaje se vyskytují například jako hrací plochy různých her, či rastrové obrázky.

**Výpis tabulky:**

Výpist tabulky do shellu je možné vnořením dvou cyklů, kdy jeden bude mít za úkol procházet položky základní tabulky a druhý položky seznamu, který je uložen v položce základní tabulky.

Procházet tabulku můžeme buď po prvkách:

**def**vypis(tab):

**for** riadok **in** tab:

**for** prvok **in** riadok:

print(prvok, end=' ')

print()

Nebo za pomocí indexů:

**def**vypis(tab):

**for** i **in**range(len(tab)):

**for** j **in**range(len(tab[i])):

print(tab[i][j], end=' ')

print()

Výpis pomocí indexů se hodí v případě, kdy potřebujeme při zpracovávání prvku pracovat i s jinými prvky, např. při porovnání přilehlých prvků kolem.

**Vytváření dvojrozměrných tabulek:**

Dvojrozměrné tabulky vytvářejí buď přímím zadáním údajů:

**>>>**matica = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

Nebo:

**>>>**riadok1 = [1, 2, 3]

**>>>**riadok2 = [4, 5, 6]

**>>>**riadok3 = [7, 8, 9]

**>>>**matica = [riadok1, riadok2, riadok3]

Ale častěji se používá nějakých cyklů a příkazu přiřazení (buď '=', nebo append()):

**>>>**matica = []

**>>>for** i **in**range(3):

matica.append([0, 0, 0])

Nikdy ale ne násobením první tabulky, protože takto bychom vytvořili jen více odkazů na stejnou tabulku:

**>>>**matica1 = [[0, 0, 0]] \*3

Nicméně tento zápis už je v pořádku:

**>>>**matica2 = [[0] \*3, [0] \*3, [0] \*3]

Protože takto vytvoříme kopii položky v tabulce.

Někdy se k vytvoření tabulky používá tato funkce, kde určíte počet řádků a sloupců a případně zda zde chcete mít jinou hodnotu než '0':

**def**vyrob(pocet\_riadkov, pocet\_stlpcov, hodnota=0):

vysl = []

**for** i **in** range(pocet\_riadkov):

vysl.append([hodnota] \* pocet\_stlpcov)

**return** vysl

Anebo takto:

**def**vyrob(pocet\_riadkov, pocet\_stlpcov, hodnota=0):

vysl = [**None**] \* pocet\_riadkov *# None alebo ľubovoľná iná hodnota*

**for** i **in** range(pocet\_riadkov):

vysl[i] = [hodnota] \* pocet\_stlpcov

**return** vysl

**Změnění prvků v tabulce:**

K prvkům tabulky se dostaneme za pomocí indexů, kdy první index odkazuje na položku v první tabulce a druhý index odkazuje na položku ve vnořené tabulce. Samotné prvky měníme příkazem přiřazení '='.

Měnit obsah tabulky můžeme také pomocí procházení cyklem.

Zde je funkce, která očísluje vzestupně všechny položky:

**def** ocisluj(tab):

poc =0

**for** i **in** range(len(tab)):

**for** j **in** range(len(tab[i])):

tab[i][j] = poc

poc +=1

Kterou můžeme zapsat i takto:

**def**ocisluj(tab):

poc =0

**for** riadok **in** tab:

**for** j **in**range(len(riadok)):

riadok[j] = poc

poc +=1

**Zobrazení dvojrozměrné tabulky v grafické ploše:**

**import tkinter**

**def** kresli\_text(tab):

d =20

**for** r, riadok **in** enumerate(tab):

**for** s, prvok **in** enumerate(riadok):

canvas.create\_text(s\*d +10, r\*d +10, text=prvok)

canvas = tkinter.Canvas()

canvas.pack()

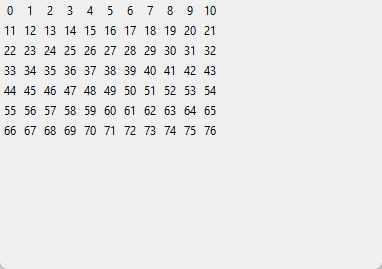
t =vyrob(7, 11)

ocisluj(t)

kresli\_text(t)

tkinter.mainloop()

Výsledek:

[](http://python.input.sk/_images/13_01.png)

Namísto čísel můžeme na jejich místo vykreslit barevné čtverečky:

**import tkinter**

**def** kresli(tab, d=20):

farby = ('white', 'black', 'red', 'blue')

**for** r, riadok **in** enumerate(tab):

**for** s, prvok **in** enumerate(riadok):

x, y = s\*d +5, r\*d +5

farba = farby[prvok %len(farby)]

canvas.create\_rectangle(x, y, x+d, y+d,

fill=farba, outline='light gray')

canvas = tkinter.Canvas()

canvas.pack()

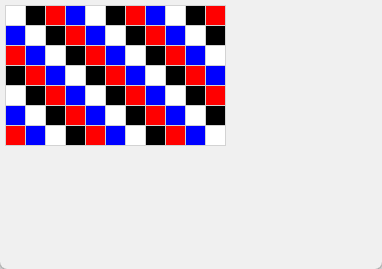
t =vyrob(7, 11)

ocisluj(t)

kresli(t)

tkinter.mainloop()

Výsledek:

[](http://python.input.sk/_images/13_02.png)

Vyrobenou síť je možné libovolně zabarvit, např:

canvas = tkinter.Canvas()

canvas.pack()

n =11

t =vyrob(n, n)

**for** i **in** range(n):

**for** j **in** range(n):

**if** i ==0**or** i == n-1**or** j ==0**or** j == n-1:

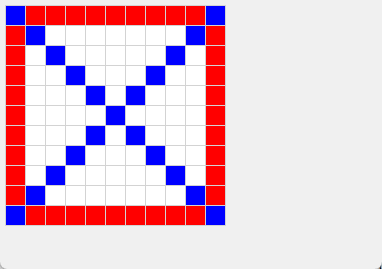
t[i][j] =2

t[i][i] = t[i][n-1-i] =3

kresli(t)

tkinter.mainloop()

Vyrobí toto:

[](http://python.input.sk/_images/13_04.png)

**Hodnota 'None':**

každá funkce ukončená bez 'return' vrací 'None'. Stejně tak vrací None i funkce s 'return' pokud žádnou hodnotu neobdržela, nebo pokud má přímo uvedené, aby vrátila tuto hodnotu, např:

**return None**

Tuto hodnotu můžeme často využívat v situacích, když chceme oznámit, že hledání výsledku bylo neúspěšné.

Za pomocí této hodnoty můžeme i testovat, zda se určitá hodnota vrací, či vevrací:

vysledok =index(tab, hodnota)

**if** vysledok **is None**:

print('nenasiel')

**else**:

riadok, stlpec = vysledok

Je dobré upřednostnit v takovémto ověření zápis s 'is None', před zápisy '== None' a '!= None'.

**Tabulky s různě dlouhými řádky:**

Dvojrozměrné tabulky nemusí být vždy symetrické, ale každý řádek může obsahovat různě velkou tabulku.

Funkce, která nám vrátí seznam délky jednotlivých řádků:

**def** dlzky(tab):

vysl = []

**for** riadok **in** tab:

vysl.append(len(riadok))

**return** vysl

Funkce pro výrobu tabulky s různně dlouhými řádky:

**def** vyrob\_d(dlzky, hodnota=0):

vysl = []

**for** dlzka **in** dlzky:

vysl.append([hodnota] \* dlzka)

**return** vysl

Ukázka programu, který vytvoří tabulku v grafické ploše a po té každý řádek zkrátí o náhodnou hodnotu :

**importrandom**

canvas = tkinter.Canvas()

canvas.pack()

n =11

t =vyrob(n, n) *# tabuľka n x n samých 0*

**for** riadok **in** t:

**for** i **in** range(n):

riadok[i] = random.randint(0, 2) *# všetky prvky sú náhodné z <0, 2>*

kresli(t)

canvas.after(1000)

**for** i **in** range(n):

t[i] = t[i] [:random.randrange(n)] *# náhodné generování délky tabulky*

kresli(t)

tkinter.mainloop()

**Pár příkladů práce s dvojrozměrnými tabulkami:**

Zvýšení všech prvků o jeden:

**def** zvys\_o\_1(tab):

**for** riadok **in** tab:

**for** i **in**range(len(riadok)):

riadok[i] +=1

Vytvoření nové tabulky, kopií a zvýšením hodnot:

**def** o\_1\_viac(tab):

nova\_tab = []

**for** riadok **in** tab:

novy\_riadok =list(riadok) *# kópia pôvodného riadka*

**for** i **in** range(len(novy\_riadok)):

novy\_riadok[i] +=1

nova\_tab.append(novy\_riadok)

**return** nova\_tab

Kopie dvojrozměrné tabulky:

**def** kopia(tab):

nova\_tab = []

**for** riadok **in** tab:

nova\_tab.append(list(riadok))

**return** nova\_tab

Číslování tabulky po sloupcích (tabulka musí mít řádky stejně dlouhé:

**def**ocisluj\_po\_stlpcoch(tab):

poc =0

**for** j **in** range(len(tab[0])):

**for** i **in** range(len(tab)):

tab[i][j] = poc

poc +=1

Spočítání počtu výskytů nějaké hodnoty za pomocí metody count():

**def**pocet(tab, hodnota):

vysl =0

**for** riadok **in** tab:

vysl += riadok.count(hodnota)

**return** vysl

Zjištění, či je nějaká matice (dvojrozměrný seznam) symetrická:

**def** symetricka(matica):

**for** i **in** range(1, len(matica)):

**for** j **in** range(i):

**if** matica[i][j] != matica[j][i]:

**return False**

**return True**

Funkce vracící pozici prvního výskytu nějaké hodnoty:

**def** index(tab, hodnota):

**for** i **in** range(len(tab)):

**for** j **in** range(len(tab[i])):

**if** tab[i][j] == hodnota:

**return** i, j

**Hra Life:**

V nekonečné čtvercové síti žijí bunky, které se různě rozmnožují a umírají podle následujících pravidel:

1) Když pole má kolem sebe 2 nebo 3 sousedy s hodnotou 1 (černě zabarvené), buňka přežívá, či ožívá do další generace.

2) Když pole nesplňuje první podmínku, buňka dostává v další generaci hodnotu 0 (bíle zabarvená).

**import tkinter**

**import random**

**def** nahodne(n):

vysl = []

**for** i **in**range(n):

vysl.append([])

**for** j **in**range(n):

vysl[-1].append(random.randrange(2))

**return** vysl

**def**kresli(tab, d=8):

canvas.delete('all')

**for** r, riadok **in** enumerate(tab):

**for** s, prvok **in** enumerate(riadok):

x, y = s\*d +5, r\*d +5

farba = ('white', 'black')[prvok]

canvas.create\_rectangle(x, y, x+d, y+d, fill=farba, outline='lightgray')

canvas.update()

**def** nova\_generacia(p):

nova = []

**for** r **in** range(len(p)):

nova.append([0] \*len(p[r]))

**for** r **in** range(1, len(p)-1):

**for** s **in** range(1, len(p[r])-1):

ps = (p[r-1][s-1] + p[r-1][s] + p[r-1][s+1] +

p[r][s-1] + p[r][s+1] +

p[r+1][s-1] + p[r+1][s] + p[r+1][s+1])

**if** ps==3 **or** ps==2 **and** p[r][s]:

nova[r][s] =1

**return** nova

canvas = tkinter.Canvas(width=600, height=600)

canvas.pack()

plocha = nahodne(50)

kresli(plocha)

**while** True:

plocha = nova\_generacia(plocha)

kresli(plocha)

tkinter.mainloop()