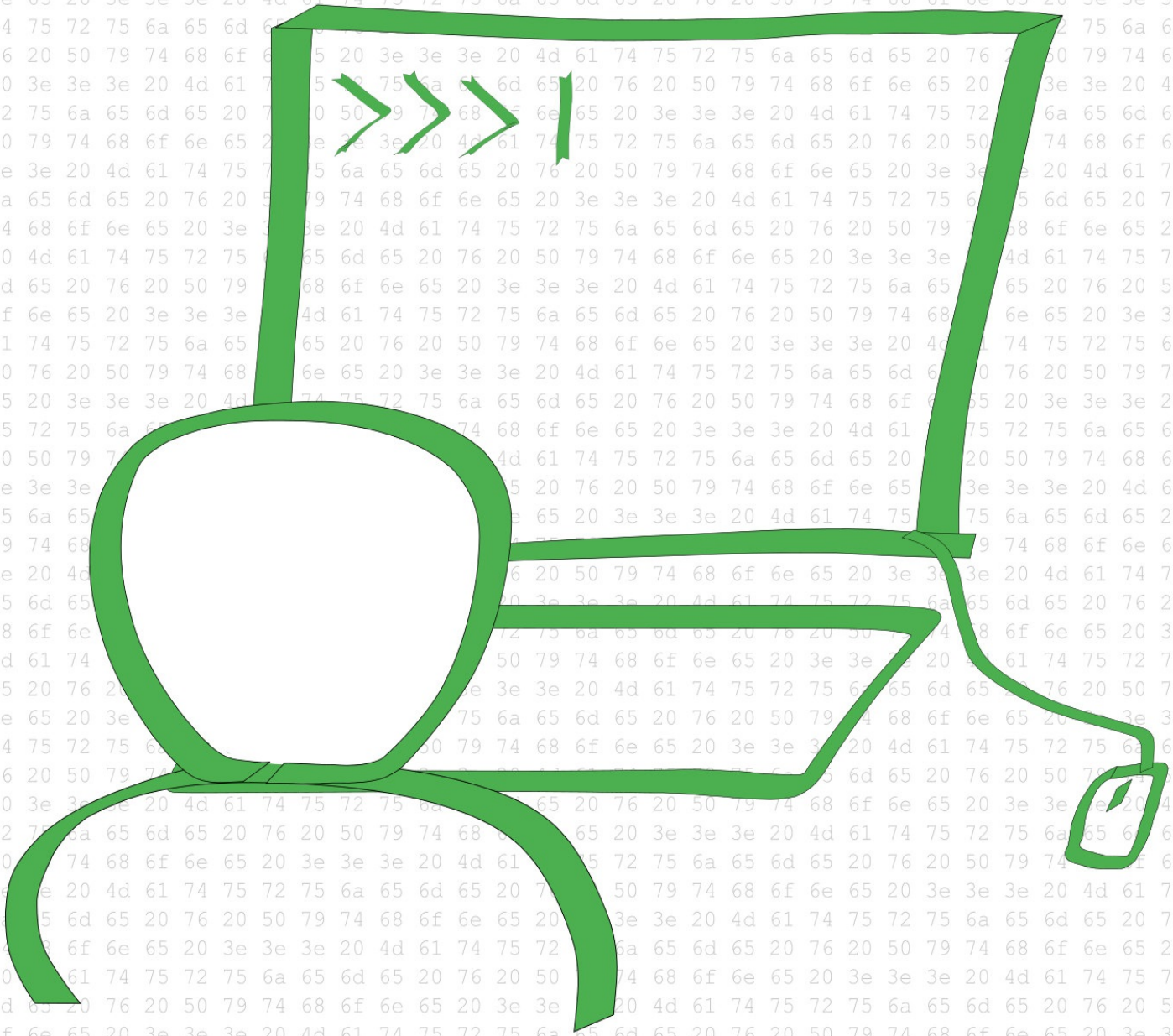
Maturujeme v Pythone

zbierka riešených úloh k maturite z informatiky

Peter Kučera, Jaroslav Výbošťok



www.programujemevpythone.sk

Maturujeme v Pythone

zbierka riešených úloh k maturite z informatiky

Jazyková korektúra: Mgr. Katarína Kučerová

Prvé vydanie, 2018

Verzia číslo: 20180301

Vydavateľ: Mgr. Peter Kučera

Ukážka z e-knihy Maturujeme v Pythone

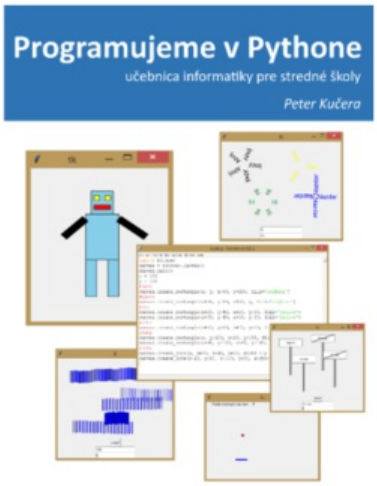
Zbierku a ďalšie materiály si môžete zakúpiť aj priamo na stránkach autora:
http://www.programujemevpythone.sk/ a https://www.facebook.com/programujemevpythone

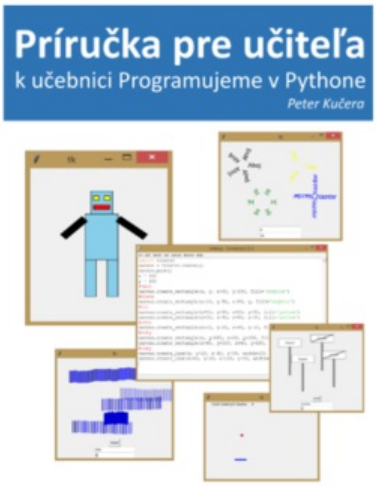
ISBN 978-80-972987-2-2 (pdf)

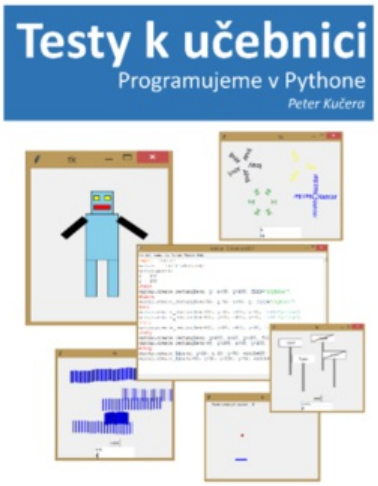
ISBN 978-80-972987-3-9 (epub)

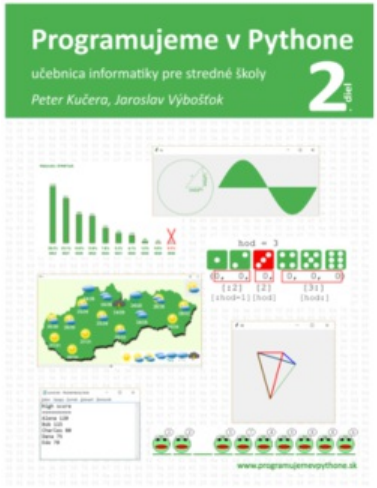
ISBN 978-80-972987-4-6 (mobi)

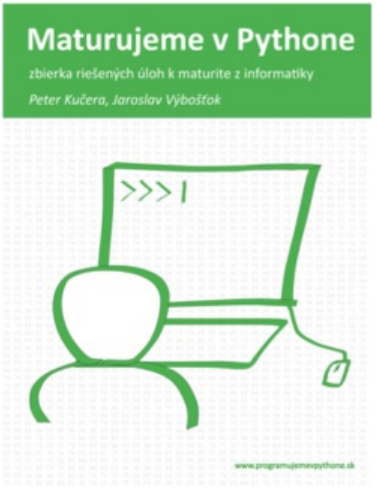
Z NAŠEJ PONUKY











www.programujemevpythone.sk
www.facebook.com/programujemevpythone

Obsah

Úvod

Malcolm Gladwell vo svojej knihe Výnimoční uvádza, že ak chce človek dosiahnuť úspech v istej oblasti, musí trénovať alebo odpracovať 10 000 hodín. Autor sa snaží pravidlom poukázať na to, že samotný talent nestačí alebo nie je nutný na dosiahnutie úspechu. Je však potrebné vynakladať množstvo úsilia a systematicky trénovať. Niektorí psychológovia jeho tvrdenie odmietajú.

Či už je tvrdenie pravdivé alebo nie, myslíme si, že prax nám pomáha zlepšovať sa v danej oblasti. Veríme, že týchto 64 úloh vám pomôže zlepšiť sa v programovaní a získať potrebnú prax, ktorú ocení nielen maturitná komisia, ale aj vy v ďalšom štúdiu či v práci.

Ak ste už pri programovaní zažili situáciu, že vás pohltilo natoľko, že ste zabudli na plynutie času a všetkého okolo, nepociťovali ste hlad, smäd ani únavu, dostali ste sa do stavu flow. Flow pomenoval a definoval psychológ Mihaly Csikszentmihalyi.

Dôležitým predpokladom stavu flow je, že náročnosť zadanej úlohy je primeraná zručnostiam. Keď sa naše zručnosti zlepšujú a s nimi sa primerane zvyšuje náročnosť úloh, nastáva pravé zaujatie a pocit zmysluplnosti – teda flow.

Nie je nutné, aby ste vyriešili všetky úlohy a v danom poradí. Vyberajte si ich tak, aby vás bavili a boli pre vás primeranou výzvou, nech vás programovanie pohltí a baví :)

Prajeme vám príjemné chvíle a flow pri riešení úloh, ale aj výborný výsledok na maturitnej skúške.

Peter Kučera, Jaroslav Výbošťok

O zbierke úloh a maturite z informatiky

Priebeh maturitnej skúšky

Maturitnú skúšku z informatiky tvorí ústna odpoveď študenta pred trojčlennou maturitnou komisiou. Študent si vyžrebuje jedno zo schválených maturitných zadaní. Zadanie obsahuje dve úlohy. Prvá úloha je z programovania a druhá úloha je zo základov informatiky. Celkový čas prípravy na obe úlohy je 30 minút. Študent si určuje, ako si čas prípravy rozdelí medzi jednotlivé úlohy. Po príprave nasleduje ústna odpoveď, ktorá trvá 20 minút.
V tejto zbierke nájdete prvé (programátorské) úlohy maturitných zadaní.
Čo všetko má študent vedieť na maturitu, určujú Cieľové požiadavky na vedomosti a zručnosti maturantov z informatiky, ktoré sú zverejnené na stránke Štátneho pedagogického ústavu (www.statpedu.sk).

Čo obsahuje táto e-kniha?

Zbierka obsahuje 64 úloh (34 grafických a 30 konzolových), ktoré svojou náročnosťou a zložením spĺňajú požiadavky na prvú časť (úlohu) maturitného zadania z informatiky (algoritmická úloha). V prvej časti zbierky sa nachádza zadanie úloh. Každá úloha má na konci odkaz na jej riešenie (riešenie obsahuje odkaz na zadanie), ktoré je v druhej časti zbierky. Niektoré úlohy obsahujú aj viaceré alternatívy riešenia. Pri zadaniach alebo riešeniach sa môžu nachádzať aj doplňujúce otázky, ktoré vás majú motivovať k premýšľaniu a pomôcť vám rozvíjať kritické myslenie a objavovanie súvislostí. Na konci riešenia každej úlohy sú symboly, ktoré popisujú, ktoré poznatky sú v riešení úlohy využité (napr.: textové reťazce, zoznamy, n-tice, funkcie a pod.). Netvrdíme, že bez týchto poznatkov sa úlohy nedajú vyriešiť.
Na začiatku zbierky nájdete prehľad úloh spolu s popisom poznatkov, ktoré môžu byť na riešenie úlohy potrebné.

Ako riešiť úlohy?

Úlohy môžete riešiť v ľubovoľnom poradí. Ani ich umiestnenie v zbierke nemá kritérium. Úlohu by ste mali vyriešiť približne za 20 až 30 minút (čas z prípravy na maturite + čas z odpovede). Nie je nutné, aby ste úlohu na maturite vyriešili už počas prípravy, môžete ju dokončiť aj v priebehu odpovede. Náročnosť úloh zodpovedá náročnosti prvej úlohy maturitných zadaní. Netvrdíme, že všetky úlohy majú rovnakú náročnosť, no sú v istých medziach náročnosti prvej úlohy maturitného zadania. Môže sa stať, že niektorá úloha sa vám zdá veľmi ľahká a iná veľmi ťažká, no môže to byť spôsobené len kontextom úlohy, na ktorý ste doposiaľ neboli zvyknutí, alebo ste práve úlohy s podobným kontextom trénovali.
Odporúčame vám zapisovať si do tabuľky dátum riešenia úlohy, celkový čas na jej vyriešenie a nejaké poznámky. Zo zápisov v tabuľke by ste mali vidieť, že sa postupne znižuje čas potrebný na vyriešenie úlohy, čo znamená, že získavate potrebnú prax. Vhodné je tieto úlohy riešiť aj priamo na vyučovaní. V rámci skupiny môžete mať spoločnú zdieľanú tabuľku (napríklad cez Google Drive), kde si informácie o vyriešených úlohách zapisuje každý študent zo skupiny (vo svojich troch stĺpcoch). Takto budete vedieť odhadnúť časovú náročnosť úlohy aj podľa údajov od svojich spolužiakov, alebo budete vidieť, ako rýchlo ste ju v porovnaní s nimi vyriešili vy.

Skladba úlohy

V prvej úlohe maturitného zadania (programátorskej) má byť definovaný cieľ, ktorý má študent dosiahnuť vytvorením programu v konkrétnom programovacom jazyku. V zadaní úlohy nemajú byť prezradené prostriedky, ktorými sa cieľ má dosiahnuť. Výber prostriedkov si určuje študent a ich správny výber je súčasťou hodnotenia. To znamená, že v úlohe nemá byť prezradené, či sa na riešenie má použiť napríklad cyklus a ktorý cyklus, a tiež ktorý typ premennej alebo dátovej štruktúry sa má v programe využiť.
Úlohy v zbierke sú členené na menšie podúlohy (alebo nejaké vlastnosti programu) a tie sú zapísané v odrážkach. Často sú tieto odrážky gradovanými čiastkovými podúlohami celej úlohy. Toto členenie má pomôcť aj slabšiemu študentovi vyriešiť aspoň časť úlohy. Napríklad v úlohe s čítaním textového súboru môže byť v prvej odrážke prečítanie prvého riadku súboru a jeho vypísanie, čím študent preukáže, že vie minimálne správne

otvoriť vstupný súbor a prečítať prvý riadok. Až podúloha v ďalšej odrážke môže požadovať prečítanie a spracovanie celého súboru. Tiež sa môže stať, že prvá odrážka popisuje podmnožinu úlohy, ktorú treba spraviť v ďalšej odrážke. V niektorých takýchto zadaniach, ak vieme, stačí vyriešiť náročnejšiu časť popísanú v ďalšej odrážke (čím preukážeme aj vyriešenie elementárnejšej úlohy, o ktorú sa zachytíme v prípade, že náročnejšiu časť nevieme vyriešiť).
Ak sa v úlohe pracuje s textovým súborom, jeho ukážku nájdete priamo v zadaní. Tieto vstupné súbory sú pripravené ako súčasť zbierky. Ich názov je rovnaký ako je v zadaní. Riešenie úloh nájdete aj v samostatných súboroch (`názov_úlohy.py`), sú priložené k e-knihe.

Čo ešte v našich učebniciach nezaznelo a nájdete v riešení niektorých úloh?

V riešení niektorých úloh sme použili aj funkcie, ktoré sme nepoužili v našich učebniciach. Napriek tomu sme ich zaradili do riešenia, aby ste sa aj pri tomto opakovaní naučili aj nejaké nové drobnosti. Tu zhrnieme niektoré z nich:
`canvas.create_polygon(x1, y1, x2, y2, .... xn, yn)` - príkaz kreslí uzavreté vyplnené útvary. Je veľmi podobný kresleniu čiar canvas.create_line(), rozdiel je len v tom, že čiarou automaticky spojí posledný nakreslený bod s prvým a vnútro útvaru vyplní nastavenou farbou (`fill='farba'`).
Dvojicu príkazov:

```
button1 = tkinter.Button(text='popis', command=funkcia)
button1.pack()
```

môžeme skrátene zapísať aj takto:

```
tkinter.Button(text='popis', command=funkcia).pack()
```

Pri odchytávaní udalostí na stlačenie klávesnice nemusíme použiť príkaz `canvas.bind_all()`, stačí aj `canvas.bind()`. Pred tým aktivujeme odchytávanie klávesnice priamo z canvasu pomocou funkcie `canvas.focus_set()`. Postačí nám teda `bind`, nepotrebujeme odchytávať udalosť z ktorýchkoľvek iných widgetov (súčiastok) v okne aplikácie.

Prehľad všetkých úloh

```
61. Delenie_____________________________'', (), Cnv, fx()
62. Anketa______________________________'', [], txt, Cnv, fx()
63. Dopravný prieskum___________________(), [], txt
64. Zasadací poriadok___________________'', (), [], txt, Cnv, fx()
```

Prehľad grafických úloh

```
1. Háďa_________________________________[], Cnv, aft.
3. Displej v električke_________________'', [], txt, Cnv, fx(), aft.
6. Noty_________________________________'', txt, Cnv, fx(), aft.
8. Nájdi puknutý tanier_________________'', [], Cnv, fx()
9. Preteky lodičiek_____________________'', [], Cnv, fx()
10. Vykreslenie križovky 1______________'', (), [], txt, Cnv, fx()
13. Žrút________________________________'', (), [], Cnv
15. Zobrazenie zátvoriek________________'', [], Cnv
18. Kresliaci robot 1____________________'', [], Cnv
19. Preklopenie obrázka_________________'', [], [[]], txt, Cnv
26. Krajina______________________________[], Cnv, fx()
30. NIM_________________________________'', Cnv, fx()
31. Čiernobiely obrázok_________________'', txt, Cnv, fx()
33. Spektrum odtieňov___________________'', [], txt, Cnv
35. Vykreslenie komprimovaného obrázka_____[], txt, Cnv, fx()
36. Vykreslenie križovky 2______________'', (), [], txt, Cnv, fx()
41. Výber jedla_________________________'', [], txt, Cnv
42. Zachráň padajúce vajíčko____________'', Cnv, fx(), aft.
43. Pyrotechnik_________________________[], Cnv, fx(), aft.
44. Uhádni padajúce slovo_______________'', Cnv, fx(), aft.
47. Čiarový kód_________________________'', txt, Cnv, fx()
49. Rezervácia miesteniek_______________'', [], txt, Cnv, fx()
50. Obrys obrázka_____________________txt, Cnv
53. Zástavba na ulici____________________[], txt, Cnv
54. Lodičky_____________________________[[]], txt, Cnv, fx()
55. Kresliaci robot 2____________________'', [], txt, Cnv, fx()
56. Editor levelov 1_____________________[], txt, Cnv, fx()
57. Editor levelov 2_____________________[], Cnv, fx()
58. Trasa linky metra___________________'', [], txt, Cnv
59. Vyťaženosť autobusovej linky________'', [], txt, Cnv, fx()
60. Kalkulačka__________________________'', Cnv, fx()
61. Delenie_____________________________'', (), Cnv, fx()
62. Anketa______________________________'', [], txt, Cnv, fx()
64. Zasadací poriadok___________________'', (), [], txt, Cnv, fx()
```

Prehľad konzolových úloh

```
2. Háďa - priebeh hry__________________'', txt, fx()
4. Záznamy z meteorologických staníc___'', (), [], txt
5. Označené jedlá______________________'', {}, txt
7. Náhodné skúšanie____________________'', []
11. Súťaž v behu_______________________(), [], txt
12. Vyťaženosť liniek dopravného podniku_____'', [], txt
14. Skok do diaľky_____________________(), [], {}, txt
16. Mená v stĺpcoch____________________'', [], txt
17. Tajná tabuľka______________________'', []
20. Tabuľka početností_________________'', [], txt
21. Poprehadzovaný text 1______________'', [], txt
22. Poprehadzovaný text 2______________'', [], txt
23. Zašifrovaný text 1_________________'', txt, fx()
24. Zašifrovaný text 2_________________'', txt, fx()
25. Lotéria____________________________'', [], txt, fx()
27. Vírus_______________________________'', [], txt, fx()
28. Hlasovanie 1_______________________'', (), [], txt
```

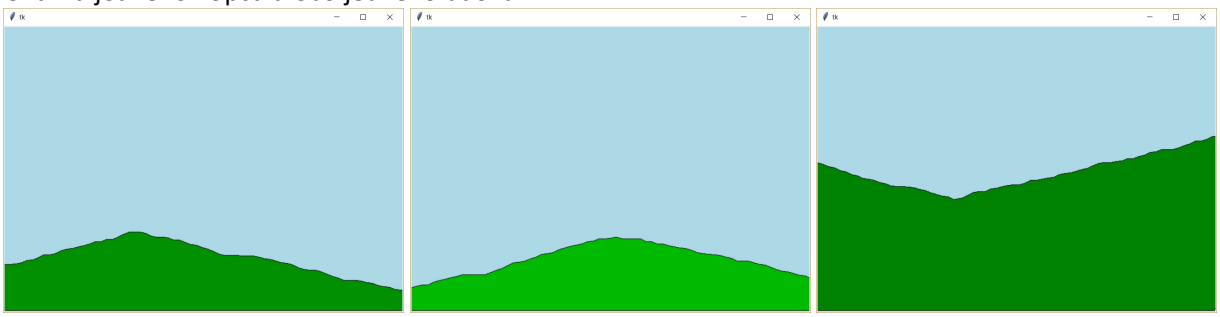
```
29. Kompresia obrázka_____________________'', txt, fx()
32. Konverzia súboru 1____________________'', txt, fx()
34. Dekompresia obrázka___________________'', [], txt, fx()
37. Hlasovanie 2__________________________'', [], txt
38. Spokojnosť zákazníkov 1_______________'', [], txt
39. Spokojnosť zákazníkov 2_______________'', [], txt
40. Analýza údajov________________________'', [], txt
45. Učenie sa slovíčok____________________'', [], txt
46. Obesenec______________________________'', [], txt
48. Násobilka_____________________________'', (), [], txt, fx()
51. Kompresia oznamu 1____________________'', []
52. Kompresia oznamu 2____________________'', []
63. Dopravný prieskum_____________________(), [], txt
```

26 Krajina

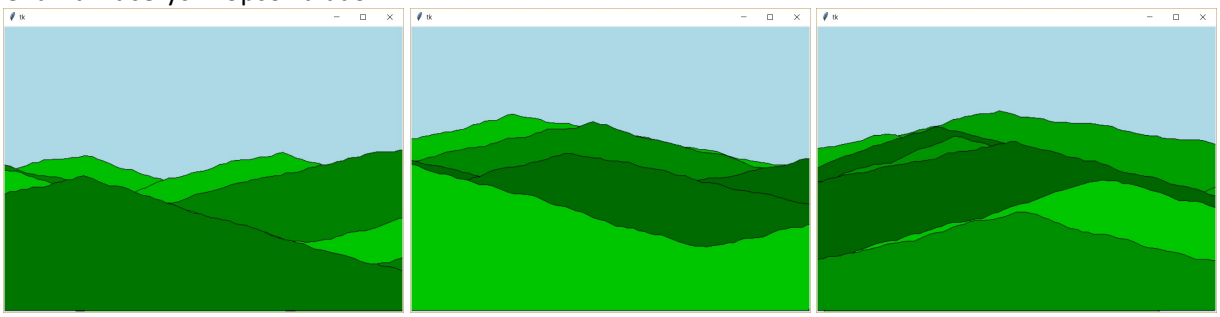
Pomocou programu môžeme generovať a vykreslovať náhodnú krajinu. Vytvorte program, ktorý:

- vygeneruje údaje pre kopec, pričom si najprv náhodne určí x-ovú pozíciu vrcholu a y-ovú súradnicu počiatočnej výšky kopca. Pre kopec platí, že jeho výška je v prvej časti (pred vrcholom) neklesajúca a za vrcholom nerastúca. Zmena v reliéfe kopca môže byť každých 10 bodov a je náhodná voči predchádzajúcemu stavu výšky.
- vykreslí kopec pomocou príkazu `canvas.create_polygon()`. Farba kopca je niektorý náhodný odtieň zelenej farby.
- sa náhodne rozhodne, či ide kresliť kopec alebo údolie (či je najprv neklesajúca postupnosť a potom nerastúca alebo opačne) a vykreslí jeden kopec alebo jedno údolie.
- opakovane vykreslí viac náhodných kopcov / údolí, čím vznikne vygenerovaná krajina,
- po stlačení medzery nakreslí novú sériu náhodných kopcov a údolí.

Ukážka jedného kopca alebo jedného údolia:



Ukážka viacerých kopcov a údolí:



Otázky:

1. Navrhnite formát textového súboru na uloženie údajov nakreslenej a vygenerovanej krajiny.
2. Dá sa z ukážky určiť, koľko kopcov (resp. údolí) program vygeneroval? Zdôvodnite svoju odpoveď.

riešenie úlohy

40 Analýza údajov

Lokálne potraviny sa rozhodli, že zistia spokojnosť s poskytnutými službami u svojich zákazníkov. Pri východe z predajne nainštalovali box, v ktorom zákazník vyjadrí svoju spokojnosť / nespokojnosť pomocou dotykovej obrazovky. Všetky vyjadrenia sa zapíšu do textového súboru. Vyjadrenia sú v textovom súbore `spokojnost_1.txt` (k dispozícii sú aj súbory `spokojnost_2.txt` a `spokojnost_3.txt`). Na každom riadku je jedno vyjadrenie zákazníka. Vyjadrenie obsahuje čas zaznamenania v tvare `hodina:minúta`, potom nasleduje jedna medzera a text `áno` alebo `nie` podľa toho, či bol zákazník spokojný alebo nespokojný. V súbore sú vyjadrenia z viacerých dní.
Ukážka vstupného textového súboru:

```
15:38 áno
15:39 áno
14:33 áno
08:38 áno
07:42 áno
15:20 áno
```

Ukážka výstupu:

```
1. deň - počet reakcií:2
2. deň - počet reakcií:1
3. deň - počet reakcií:1
4. deň - počet reakcií:2
Počet všetkých vyjadrení: 6
Hodina:7 Reakcií zákazníkov:1
Hodina:8 Reakcií zákazníkov:1
Hodina:14 Reakcií zákazníkov:1
Hodina:15 Reakcií zákazníkov:3
Počet dní: 4
```

Vytvorte program, ktorý zistí a vypíše:

- celkový počet vyjadrení,
- počet všetkých vyjadrení v jednotlivých hodinách dňa, ale iba v tých hodinách, keď boli nejaké vyjadrenia,
- počet dní, počas ktorých sa zbierali vyjadrenia (predpkladajme, že každý deň bolo zaznamenané aspoň jedno vyjadrenie a že vstupné dáta sú zapísané v takom poradí, ako boli zrealizované),
- počet vyjadrení v jednotlivých dňoch.

riešenie úlohy

26 Krajina

```
import tkinter, random
canvas = tkinter.Canvas(width=700, height=500, bg='lightblue')
canvas.pack()

def kresli_kopec():
    kopec = []
    smer = random.choice((1,-1))                #  {riadok A}
    kopec.append(0)
    kopec.append(random.randint(200, 500))
    vrchol = random.randint(100, 600)
    for i in range(vrchol // 10):
        nova_hodnota = kopec[-1] + smer * random.randint(0, 5)
        kopec.append(i*10+1)
        kopec.append(nova_hodnota)
    smer = -1 * smer
    for i in range((700-vrchol) // 10 + 10):
        nova_hodnota = kopec[-1] + smer * random.randint(0, 5)
        kopec.append(i*10+vrchol)
        kopec.append(nova_hodnota)

    kopec = [0, 500] + kopec + [700, 500]
    farba = '#00{:02x}00'.format(random.randint(100, 200)) #  {riadok B}
    canvas.create_polygon(kopec, fill=farba, outline='black')

def prekresli(event):
    canvas.delete('all')
    for i in range(10):
        kresli_kopec()

#kresli_kopec()
canvas.bind_all('<space>', prekresli)
```

Otázky:

1. Čo spôsobí zápis na riadku `{riadok A}`? Ako inak by sme ho mohli zapísať?
2. Vysvetlite zápis na riadku `{riadok B}`.
3. Ako je vyriešená zmena stúpania medzi prvou časťou kopca a druhou časťou?
4. Koľko informácií o kopcoch je v pamäti počas behu tohto programu? Je možné optimalizovať využitie pamäte? Ak áno, ako?

zadanie úlohy

[] Cnv fx()

40 Analýza údajov

```
vyjadreni = [0] * 24

subor = open('subory/spokojnost_0.txt', 'r')
dni = 0
cas1 = '00:00'
pocet_v_dni = 0
for riadok in subor:
    riadok = riadok.strip()
    info = riadok.split()
    cas2 = info[0]
    cas2_roz = info[0].split(':')
    hodina = int(cas2_roz[0])
    minuta = int(cas2_roz[1])
    vyjadreni[hodina] += 1
    if cas2 < cas1:
        dni += 1
        print('{}. deň - počet reakcií:{}'.format(dni, pocet_v_dni))
        pocet_v_dni = 1
    else:
        pocet_v_dni += 1
    cas1 = cas2
print('{}. deň - počet reakcií:{}'.format(dni, pocet_v_dni))
pocet_vyjadreni = sum(vyjadreni)
print('Počet všetkých vyjadrení:', pocet_vyjadreni)
for i in range(24):
    if vyjadreni[i] > 0:
        print('Hodina:{} Reakcií zákazníkov:{}'.format(i, vyjadreni[i]))
print('Počet dní:', dni)
```

Otázky:

1. Podľa čoho vieme, že záznam je z ďalšieho dňa?
2. Ako funguje porovnávanie časov?

zadanie úlohy
'' [] txt