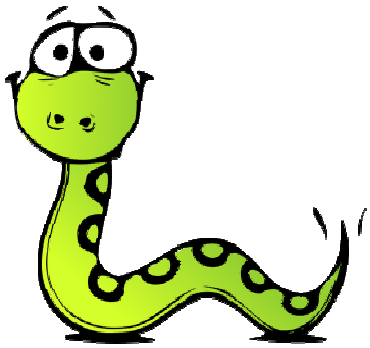
Programování v jazyce Python pro střední školy

Metodický list pro učitele

Lekce 5 – Kreslení



Andrej Blaho

Ľubomír Salanci

Václav Šimandl

Cíle lekce

- Seznámit se s grafickou plochou a s kreslením obdélníků
- Pochopit, jak funguje počítačová souřadnicová soustava
- Naučit se řešit úlohy, které vyžadují určování vzájemných pozic a rozměrů obdélníků

Dovednosti

- Zápis příkazů pro inicializaci grafické plochy
- Práce se třemi okny: textový editor s programem, grafický výstup, interaktivní konzole
- Kreslení náčrtů na papír při odvozování souřadnic

Osvojená syntaktická pravidla

- Zápisy inicializace grafické plochy (`import`, `tkinter.Canvas`, `canvas.pack()`)
- Zápis příkazu pro kreslení obdélníku (`canvas.create_rectangle`) a jeho parametrů

Průběh výuky

První i druhá úloha slouží k opakování práce s proměnnými a výpisy.

1. Už jsi směňoval koruny na eura. Teď vytvoř nový program `smena2.py`, který bude umět směnit eura na koruny. Použij proměnné `suma` a `kurz`, do kterých přiřadíš počáteční hodnoty – kolik eur chceš vyměnit a aktuální kurz (například `25.23` korun za 1 euro). Do proměnné `dostanes` přiřaď hodnotu výrazu, kterým se vypočítá, kolik korun dostaneš za svou `sumu`. Program vypíše výsledek například ve tvaru:

   ```
   Za ... eur dostaneš ... korun při kurzu ... korun za euro.
   ```

Řešení:

```
suma = 20
kurz = 25.23
dostanes = suma * kurz
print('Za', suma,
      'eur dostaneš', dostanes,
      'korun při kurzu', kurz, 'korun za euro.')
```

Parametry v příkazu `print` jsme v tomto řešení úmyslně rozepsali na více řádků, neboť takový zápis je lépe čitelný. U žáků však předpokládáme, že zapíší všechny parametry do jednoho řádku.

V případě dlouhých zápisů Python funguje tak, že otevírací závorkou začíná zápis, který se může rozložit i na více řádků, dokud nebude uzavřený příslušnou ukončovací závorkou. Tento princip funguje i v jiných situacích, nejen v příkazu `print`.

Začátečníci často zapomínají na ukončovací závorku a Python poté vypisuje chyby až na nižších řádcích, které už se zápisem nesouvisí. Je to proto, že Python nerozumí našim úmyslům, a domnívá se, že zápis má pokračovat i dále. Toto však není potřeba zatím žákům vysvětlovat.

2. Následující program pracuje s proměnnými. Urči bez použití počítače, co program vypíše:

```
km = 8
c = 30
s = km * c + 50
print('Za jízdu o délce', km, 'km s naším taxi zaplatíš',
  s, 'korun')
```

Na počítači za použití Pythonu zkontroluj, zda byla tvá domněnka správná.

Řešení – počítač vypíše:

```
Za jízdu o délce 8 km s naším taxi zaplatíš 290 korun
```

Začínáme pracovat s grafikou. Abychom žáky nezatěžovali, zvolili jsme minimální skupinu příkazů, které jsou nutné pro práci s grafikou.

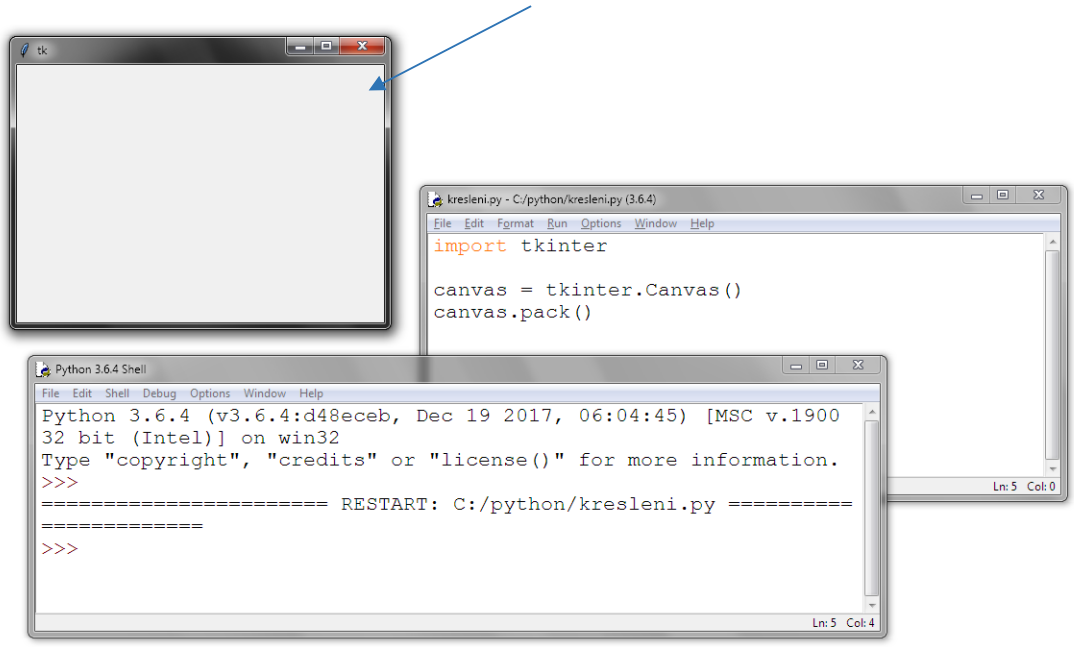
3. Doposud tvé programy počítaly a vypisovaly textové zprávy. Teď se naučíš vytvářet programy, které budou umět kreslit obrázky. Postupuj takto:

A) Vytvoř nový program `platno.py` s následujícím obsahem:

```
import tkinter

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

B) Program spusť – na obrazovce uvidíš nové okno:



C) Zjisti, zda se dá okno posouvat, měnit jeho velikost. Nakonec toto nové okno zavři.

Tvůj program vyrobil grafickou plochu `canvas`. Slovo `canvas` budeš používat i v dalších příkazech na kreslení.

Komentáře:

- V části 3.A se inicializuje grafická plocha, do proměnné `canvas` se přiřazuje objekt grafické plochy. Toto žákům nevysvětlujeme (je to příliš brzo). Tyto příkazy je potřeba opsat z pracovního listu nebo je odtud zkopírovat. Když se žáci budou dožadovat vysvětlení významu těchto příkazů, stačí jim říci, že: „tyto příkazy slouží k tomu, aby se vytvořilo prázdné okno“. Žáky ze znalosti těchto příkazů nebudeme zkoušet.
- V části 3.B může žáky zmást, že operační systém umístil okno podle svých pravidel. Někdy se tak může stát, že grafické okno bude překryté jiným oknem.
- Protože se slovo `canvas` bude v dalších zápisech často vyskytovat, můžeme se žáky diskutovat o jeho významu (např. „grafická plocha“, „malířské plátno“ apod.). Případně si při vysvětlovaní můžeme pomoci metaforou: „Windowsovské okno je rám, ve kterém se musí nacházet malířské plátno, abychom do něj mohli kreslit. Jsou ale i taková windowsovská okna, do kterých se kreslit nedá.“

Nejčastější chyby v programu, které dělají začátečníci při práci s grafikou

Čím více programátorských konstrukcí žáci poznají, tím častěji se v jejich řešení setkáme s různými typy chyb. Proto je pro nás důležité poznat, jak se projevují. Následující chyby můžeme otestovat v interaktivním režimu:

- Překlepy při zápisu složitějších konstrukcí:

```
>>> import tknter
...
ModuleNotFoundError: No module named 'tknter'
```

  Označuje překlep v názvu grafického modulu `tkinter`, opravíme:

```
>>> import tkinter
```

- Chybná velikost písmen ve slově `Canvas`:

```
>>> canvas = tkinter.canvas()
...
AttributeError: module 'tkinter' has no attribute
'canvas'
```

  Slovo `Canvas` mělo mít první písmeno velké. Python je na malá a velká písmena velmi citlivý.

- Chybějící tečka mezi slovy `tkinter` a `Canvas`:

```
>>> canvas = tkinterCanvas()
...
NameError: name 'tkinterCanvas' is not defined
```

Mezi slova `tkinter` a `Canvas` jsme měli vložit znak `.` (tečka). Python to nyní pochopil jako jedno dlouhé nesrozumitelné slovo `tkinterCanvas`.

- Chybná velikost písmen ve slově `pack`:

```
>>> canvas = tkinter.Canvas()
>>> canvas.Pack()
...
AttributeError: 'Canvas' object has no attribute 'Pack'
```

Jde o překlep ve slově `Pack`, které mělo být zapsané malými písmeny:

```
>>> canvas.pack()
```

- Chybějící závorky za slovem `Canvas`:

Chyba se hůře odhaluje, neboť když zapomeneme na tyto závorky, dostaneme velmi nesrozumitelnou zprávu až v dalším příkazu:

```
>>> canvas = tkinter.Canvas
>>> canvas.pack()
...
TypeError: pack_configure() missing 1 required
positional argument: 'self'
```

- Chybějící závorky za slovem `pack` v kódu:

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

V případě této chyby je situace ještě méně srozumitelná, neboť v tomto případě Python nehlásí žádnou chybu. Zápis `canvas.pack` je pro počítač korektním zápisem, ale neznamená to volání metody `pack()`. Důsledkem je, že se grafické okno zobrazí, ale bez plochy na kreslení. Pokud by za tímto kódem byl příkaz pro nakreslení obdélníku:

```
canvas.create_rectangle(50, 50, 150, 100)
```

(blíže o tomto příkazu viz úlohy 4 a 5), obdélník se kvůli této chybě nezobrazí.

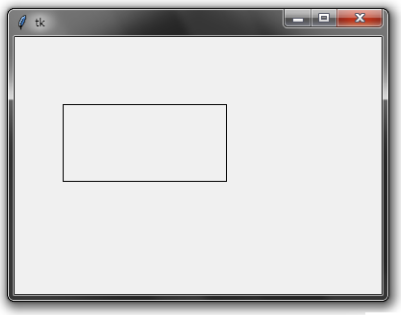
Nechme žáky, aby se postupně seznámili se zápisy příkazů.

4. Přidej do svého programu `platno.py` nový příkaz (je žlutě označený) a program opět spusť:

```
import tkinter

canvas = tkinter.Canvas()
canvas.pack()
canvas.create_rectangle(50, 70, 220, 150)
```

Uvidíš obdélník:



V této úloze jsou z pohledu žáka dvě nové věci:

- Jak funguje souřadnicová soustava (je jiná, než znají z matematiky)
- Jak funguje kreslení obdélníků (zadávají se souřadnice protilehlých vrcholů)

Pro žáka to může být kombinace dvou náročných jevů a sám nemusí přijít na to, jak to funguje. Proto následují úlohy na experimentování, které by měly žákům pomoci při objevování principu fungování:

5. V závorkách příkazu `canvas.create_rectangle( , , , )` jsou 4 čísla. Zkus je v programu `platno.py` **postupně** měnit (změny oproti předchozímu zápisu jsou zvýrazněny žlutě). Program pokaždé spusť, abys viděl, co nakreslí:

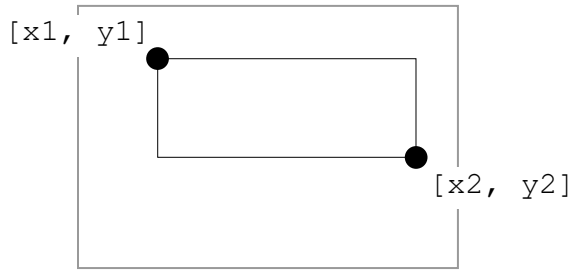
```
a) canvas.create_rectangle(0, 0, 220, 150)
b) canvas.create_rectangle(0, 0, 50, 50)
c) canvas.create_rectangle(0, 0, 250, 50)
d) canvas.create_rectangle(20, 10, 250, 50)
e) canvas.create_rectangle(20, 10, 50, 250)
```

Víš, jak tato čísla fungují?

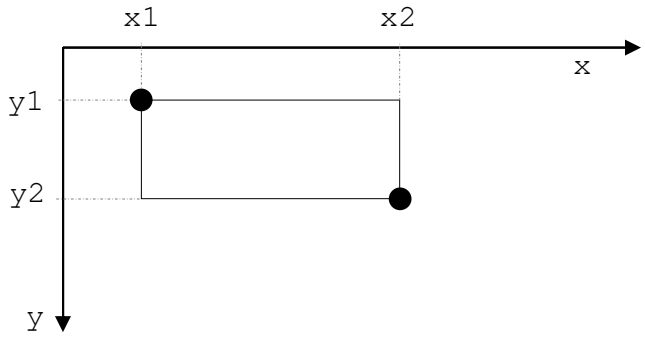
Nejdřív je potřeba rozumět, jak funguje souřadnicová soustava v počítači:

V matematice jsi zvyklý na to, že souřadnicová soustava začíná ve středu. Na počítači leží bod se souřadnicemi `[0, 0]` v levém horním rohu. Osa *x* jde zleva doprava. Osa *y* je oproti matematice převrácená – jde shora dolů (čím větší číslo, tím níže).

V příkazu `canvas.create_rectangle(`x_1`, `y_1`, `x_2`, `y_2`)` píšeme do závorek souřadnice protilehlých vrcholů obdélníku:

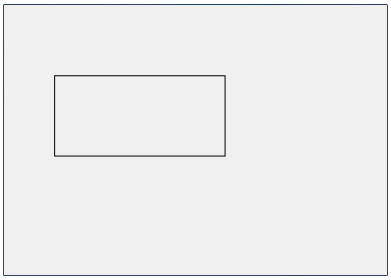


Tyto souřadnice vrcholů bychom mohli znázornit na souřadnicových osách následujícím způsobem:

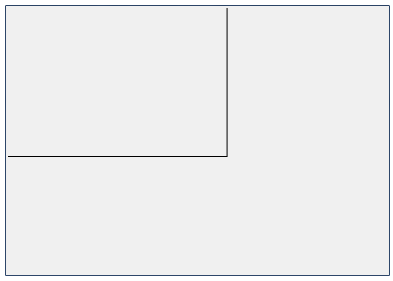


Když budou žáci experimentovat s parametry, měli by postupně vidět:

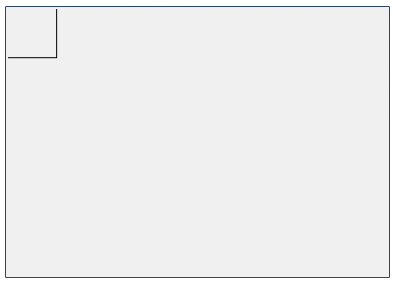
- `canvas.create_rectangle(50, 70, 220, 150)`



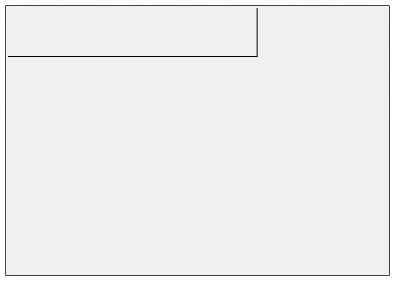
- `canvas.create_rectangle(0, 0, 220, 150)`



- `canvas.create_rectangle(0, 0, 50, 50)`



- `canvas.create_rectangle(0, 0, 250, 50)`



- `canvas.create_rectangle(20, 10, 250, 50)`

- `canvas.create_rectangle(20, 10, 50, 250)`

Když to bude potřeba, je vhodné souřadnicovou soustavu, příkazy a obdélníky postupně kreslit na tabuli. Během společné diskuze by měli žáci objevit fungování souřadnicové soustavy počítače a příkazu `create_rectangle`. Pokud žáci postupují odlišnou rychlostí, je možné kreslení realizovat na papír s menšími skupinami žáků, kteří budou takovou pomoc potřebovat.

Uvádíme, že příkaz `create_rectangle` očekává čtveřici parametrů: nejdříve souřadnice levého horního vrcholu a potom pravého dolního vrcholu. Ve skutečnosti to může být libovolná dvojice protilehlých vrcholů. Žáky však těmito různými variantami nezatěžujeme.

Grafická plocha má standardně šířku 380 bodů a výšku 266 bodů. Tyto rozměry však žákům neprozrazujeme, v 1. úloze 6. lekce budou mít možnost je sami objevit.

Na předchozích úlohách je vidět, že grafická plocha má jakoby neviditelný okraj, neboť čáry, které procházejí například přes bod `[0, 0]`, se nekreslí (resp. se kresba ořízne tak, že nejsou vidět). Okraje grafické plochy se dají změnit, aby byly vidět i tyto body, ale ani tomuto technickému detailu se prozatím nebudeme věnovat.

Následují úlohy na trénování souřadnic a parametrů, aby si žáci zvykli na počítačovou souřadnicovou soustavu a příkaz `create_rectangle`:

6. Změň svůj program `platno.py` tak, aby nakreslil obdélník, který má souřadnice protilehlých vrcholů `[50, 30]` a `[300, 200]`.

Řešení:

```
canvas.create_rectangle(50, 30, 300, 200)
```

V této lekci žáci nepracují jen se dvěma dříve používanými okny (textovým editorem s programem a interaktivní konzolí), ale seznamují se zde s dalším oknem obsahujícím grafickou plochu. Bude-li to potřeba, je vhodné se žáky diskutovat o tom, k čemu jednotlivá okna slouží. Tuto diskuzi však není vhodné realizovat na začátku lekce, kdy žáci mají jen minimum zkušeností s kreslením na plátno. Vhodnější doba je až nyní, kdy si žáci na práci s grafickou plochou částečně zvykli a více chápou její potenciál.

Následujícím příkladem začínají úlohy, ve kterých je nutné uvažovat o rozměrech obdélníku:

7. a) Spočítej bez použití počítače, jakou šířku a výšku má obdélník z předchozí úlohy.
   b*) Svou domněnku ověř za použití snímku obrazovky a libovolného grafického editoru.

Řešení – stačí takovýto výpočet:

Šířka: `300 - 50`
Výška: `200 - 30`

Když vytvoříme snímek obrazovky a rozměry odměříme v grafickém editoru, zjistíme, že rozměry jsou ve skutečnosti o jedničku větší:

Šířka: `300 - 50 + 1`
Výška: `200 - 30 + 1`

Na zohlednění ±1 v úlohách s počítáním rozměrů rozhodně nebudeme trvat (v této etapě poznávání to považujeme za nepodstatné).

8. Vytvoř nový program `obdelnik.py` a nakresli obdélník, který má jeden vrchol na souřadnicích `[200, 100]`, jeho šířka je `60` a výška `140`.

Řešení:

```
canvas.create_rectangle(200, 100, 200 + 60, 100 + 140)
```

nebo:

```
canvas.create_rectangle(200, 100, 260, 240)
```

Samozřejmě uznáme i řešení s volbou jiných protilehlých vrcholů.

9. Bez použití počítače urči a do sešitu nakresli, jak přibližně budou rozmístěné následující obdélníky. Jaká je výška a šířka každého z nich?

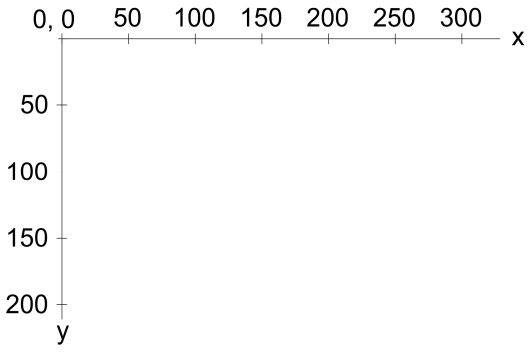
```
import tkinter

canvas = tkinter.Canvas()
canvas.pack()
canvas.create_rectangle(50, 70, 220, 150)
canvas.create_rectangle(60, 80, 130, 140)
canvas.create_rectangle(160, 90, 230, 160)
```

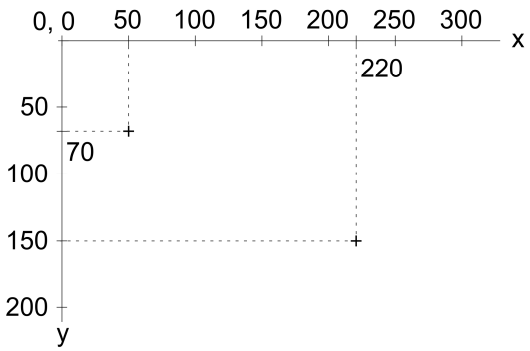
Na počítači za použití Pythonu zkontroluj, zda byla tvá domněnka správná.

Postup:

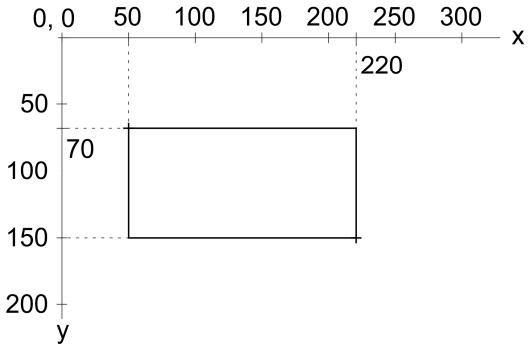
- Nakreslíme si souřadnicové osy a naznačíme souřadnice



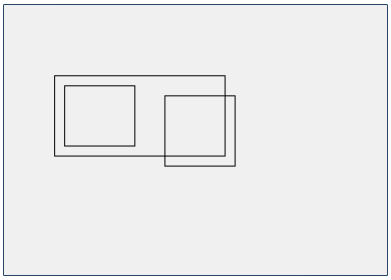
- Postupně doplníme body pro první obdélník



- Dokreslíme první obdélník

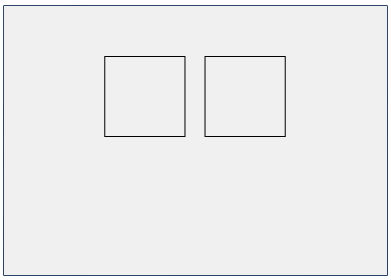


- A takto postupujeme i s ostatními obdélníky
- Nakonec naše úvahy ověříme za použití Pythonu:



Další úloha je zaměřená na kreslení čtverců – to znamená, že budou kresleny obdélníky, které mají stejnou šířku a výšku:

10. Vytvoř nový program `vedle_sebe.py`, který vedle sebe nakreslí dva čtverce se stranami délky 80 (pozici čtverců zvol podle uvážení):



Možné řešení:

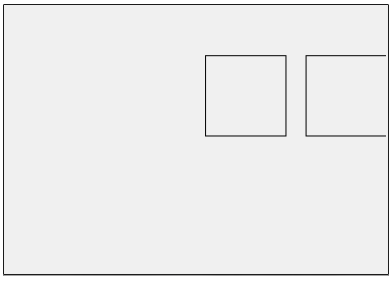
```
canvas.create_rectangle(100, 50, 100 + 80, 50 + 80)
canvas.create_rectangle(200, 50, 200 + 80, 50 + 80)
```

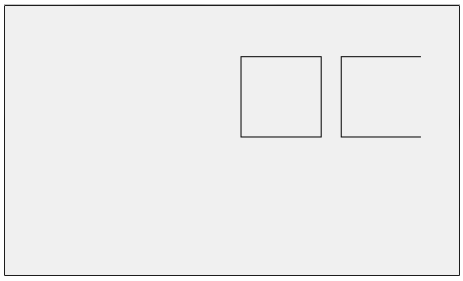
Když se budou žáci ptát, jakým příkazem se kreslí čtverec, je třeba s nimi diskutovat a přivést je na myšlenku, že mohou použít již známý příkaz.

Můžeme vidět, že v našem řešení píšeme na místech některých parametrů výrazy se součty (například `100 + 80`). Když to žáci budou zvládat, mohou jejich hodnoty vypočítat zpaměti a uvést výsledné číslo (`180`). Nevadí nám ani řešení s nevypočítanými výrazy – z nich je lepší vidět, jak souřadnice vznikají a jsou z nich dobře čitelné i rozměry obdélníku.

Pozice čtverců na plátně může být libovolná a rozhodně **netrváme** na tom, aby levý horní vrchol levého obdélníku měl souřadnice `[100, 50]` nebo aby mezi čtverci byla mezera o šířce přesně 20.

V této úloze mohou žáci rozmístit obdélníky na plátně tak, že některé z nich budou zcela nebo částečně mimo grafickou plochu (viz levý obrázek). Když následně okno s grafickou plochou zvětší, kresba zůstane stále oříznutá (viz obrázek vpravo). To je způsobeno tím, že grafická plocha zůstala po zvětšení okna stále stejně velká, jen je umístěna ve větším okně.

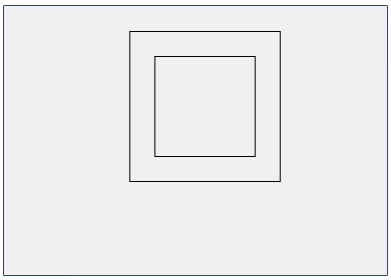




Ačkoliv je možné rozměry grafické plochy změnit, považujeme za vhodné řešení doporučit žákům, kteří se s tímto problémem setkají, aby upravili rozmístění obdélníků na ploše. Změnou velikosti grafické plochy se budeme zabývat až v metodickém listu 10. lekce, protože prozatím nechceme začátečníky těmito technickými triky zatěžovat.

Následující úloha se opět zabývá kreslením čtverců, ale problém je zkomplikovaný tím, že čtverce mají společný střed:

11. Vytvoř program `soustredne.py`, který nakreslí dva velké čtverce – jeden se stranou délky 100 a druhý 150. Čtverce budou mít společný střed jako na následujícím obrázku:



Úlohu lze řešit vícero způsoby:

- Na nějaké pozici nakreslíme větší čtverec a menší do něj umístíme tak, aby mezi nimi byla mezera `25`:

```
canvas.create_rectangle(100, 50, 250, 200)
canvas.create_rectangle(100 + 25, 50 + 25, 250 - 25, 200 - 25)
```

- Na nějaké pozici nakreslíme menší čtverec a větší nakreslíme okolo něj tak, aby mezi nimi byla mezera `25`:

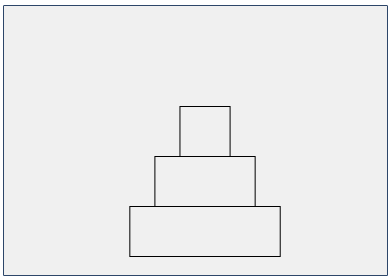
```
canvas.create_rectangle(150, 100, 250, 200)
canvas.create_rectangle(150 - 25, 100 - 25, 250 + 25, 200 + 25)
```

- Zvolíme si střed a dopočítáme souřadnice vrcholů:

```
canvas.create_rectangle(200 - 50, 100 - 50, 200 + 50, 100 + 50)
canvas.create_rectangle(200 - 75, 100 - 75, 200 + 75, 100 + 75)
```

Na úloze může být náročné odvození souřadnic vrcholů. Pokud to bude potřeba, na tabuli nakreslíme souřadnicové osy a v diskuzi se žáky dopočítáme pozice vrcholů.

12. Vytvoř program `pyramida.py`, který ze tří obdélníků o rozměrech 150x50, 100x50 a 50x50 nakreslí následující pyramidu:



Možná řešení:

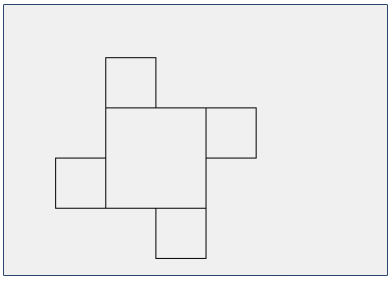
- Kreslíme od spodního největšího obdélníku. Postupujeme jako bychom obdélníky na sebe pokládali. Každý další obdélník má y-ové souřadnice vrcholů menší o 50, x-ová souřadnice levého vrcholu je zvětšená o 25 a x-ová souřadnice pravého vrcholu je zmenšená o 25:

```
canvas.create_rectangle(100, 200, 100 + 150, 200 + 50)
canvas.create_rectangle(100 + 25, 150, 100 + 150 - 25, 150 + 50)
canvas.create_rectangle(100 + 50, 100, 100 + 150 - 50, 100 + 50)
```

- Zvolíme x-ovou souřadnici středu celé stavby (například 200). Kreslíme od horního obdélníku a počítáme souřadnice protilehlých vrcholů:

```
canvas.create_rectangle(200 - 25, 100, 200 + 25, 100 + 50)
canvas.create_rectangle(200 - 50, 150, 200 + 50, 150 + 50)
canvas.create_rectangle(200 - 75, 200, 200 + 75, 200 + 50)
```

13* Vytvoř program `ornament.py`, který z pěti čtverců nakreslí následující ornament. Rozměry čtverců zvol podle svého uvážení (všechny menší čtverce budou stejně velké):



Způsobů řešení existuje vícero, my jsme zvolili následující:

Nejprve nakreslíme velký středový čtverec a poté menší čtverce v následujícím pořadí: horní čtverec, dolní čtverec, levý čtverec, pravý čtverec

```
canvas.create_rectangle(100, 100, 200, 200)
canvas.create_rectangle(100, 100 - 50, 100 + 50, 100)
canvas.create_rectangle(200 - 50, 200, 200, 200 + 50)
canvas.create_rectangle(100 - 50, 200 - 50, 100, 200)
canvas.create_rectangle(200, 100, 200 + 50, 100 + 50)
```