Středoškolská technika 2017

Setkání a prezentace prací středoškolských studentů na ČVUT

ŽELVÍ GRAFIKA (NEJEN) V PYTHONU

Linda Bulantová

Gymnázium Milevsko
Masarykova 183, Milevsko

Anotace

Cílem maturitní práce bylo vytvořit sbírku příkladů želví grafiky v Pythonu a výukovém webovém editoru studenta Masarykovy univerzity Martina Korbela. Úvodní část je věnována, zároveň je v této části uvedeno a vysvětleno množství pojmů, které jsou dále často používány. Následující kapitola je zaměřena na metodiku výuky programování s množstvím názorných příkladů.

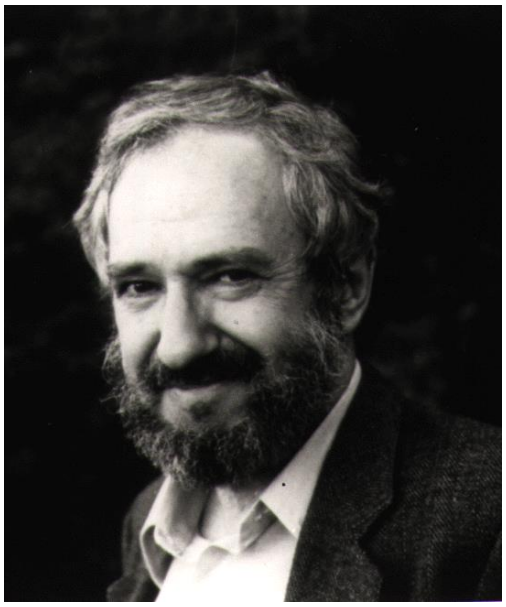
Obsah

1. Želví grafika

Informační a výpočetní technika, dnes nazývaná také informační a komunikační technologie, je na našich školách vyučována již od 5. třídy základní školní docházky. Většinou výuka zahrnuje základy práce s operačním systémem Windows, webovým prohlížečem, psaní textu v textovém editoru, kreslení jednoduchých, především bitmapových obrázků a dnes i užití sociálních sítí.. Studenti jsou seznámeni s ovládáním několika málo aplikací, ve kterých znají jen mechanický postup používání. Nerozvíjí se u nich logické myšlení, ba naopak, je po nich často vyžadováno pouhé učení se nazpaměť. Výuka programování je přitom výborný nástroj pro rozvoj logického a analytického myšlení, poskytuje studentům možnost rozvoje intelektu formou zábavy.

Od konce 70. let byla snaha přiblížit programování dětem, aby se u nich od útlého věku rozvíjelo algoritmické myšlení. Americký informatik a profesor na Massachusettském technologickém institutu Seymour Papert vymyslel jednoduchý funkcionální programovací jazyk Logo. Hlavní postavou jazyka Logo je želva (turtle), pohybující se po pláži. Když želva spustí ocásek, kreslí za sebou stopu v písku. Když ocásek zvedne, pohybuje se bez toho, že by za sebou zanechávala stopu. Způsob kreslení se nazývá želví grafika.

V své sbírce příkladů jsem se zaměřila na kreslení jednoduchých i složitějších obrázků s pomocí želví grafiky již v několika programovacích jazycích. Jedná se především o aplikaci želví grafiky a Python.



Obr. č. 1 Seymour Papert

2. Editor pro vykreslování želví grafiky

Student Masarykovy univerzity Martin Korbel vytvořil výukový webový editor pro vykreslování želví grafiky, který pracuje s jednoduchou syntaxí podobnou jazyku Logo. Celá aplikace je postavena na značkovacím jazyce HTML 5 a JavaScriptu, takže pro bezproblémové používání je třeba zvolit webový prohlížeč, který zmíněný jazyk podporuje (Chrome, FireFox, Opera). Provoz aplikace na webovém serveru není žádným způsobem omezen a nepotřebuje ani žádné dodatečné nastavení.

2.1 Popis prostředí

Obr. č. 2 Editor pro vykreslování želví grafiky

1. plátno aplikace – Místo pro vykreslování želví grafiky. Zelený trojúhelník uprostřed označuje želvu.

2. editor zdrojového kódu – Okénko editoru slouží pro vkládání příkazů. Na levé straně editoru je šedý proužek s číslováním řádků. Editor podporuje zvýrazňování syntaxe. Při krokování se v editoru zobrazí červený řádek, který značí aktuální vykonávaný kód.
3. panel ovládacích tlačítek – Tento panel bude popsán dále v této kapitole.
4. stavový řádek – Místo pro výpis aktuálního děje, v jaké fázi se vykreslování želví grafiky nachází, popřípadě informační nápovědy tlačítek.
5. panel záložek – Panel se skládá ze čtyř záložek, mezi kterými se dá přepínat.

- Chyby – V tomto panelu se zobrazují chybové hlášení informující o syntaktické chybě ve zdrojovém kódu. U každého hlášení je uvedeno i číslo řádku s chybou.
- Ladění – V případě krokování se v panelu zobrazí proměnné a jejich hodnoty. Speciální proměnná _counter označuje počítadlo zbývajících cyklů.
- Příklady – Několik hotových příkladů pro názornou ukázku aplikace.
- Syntax – Základní nápověda příkazů pro vykreslování želví grafiky. U každého příkazu jsou popsány jednotlivé parametry, případně návratové hodnoty a názorný příklad.

6. horizontální posuvník – Lišta pro úpravu velikosti plátna a editoru. Šířka aplikace se nastaví dle velikosti okna webového prohlížeče. Stisknutím a držením tlačítka myši nad touto lištou měníme poměr šířky plátna proti šířce editoru. Toto nastavení se uloží i pro budoucí použití aplikace. Při každé změně šířky se vymaže obsah plátna.
7. vertikální posuvník – Výšku plátna a editoru lze upravit stisknutím a následným držením tlačítka myši nad touto lištou. Toto nastavení se uloží i pro budoucí použití aplikace. Při každé změně výšky se vymaže obsah plátna

Obr. č. 3 Panel ovládacích tlačítek

1. Interpretuje celý kód a vykresluje kompletní obrázek želví grafiky.
2. Krokuje jednotlivé řádky zdrojového kódu a provádí pohyb želvy po plátně. Krokování je standardně prováděno automaticky v intervalu 1 krok za 350 ms. Při krokování se ve zdrojovém kódu zobrazí červený řádek, který nám určuje aktuálně vykonaný kód. Tímto tlačítkem se provádí i manuální krokování.
3. Pozastavuje automatické krokování. Opětovným stiskem ikonky děj pokračuje
4. Trvale přerušuje krokování aplikace.
5. Přepíná mód krokování z automatického (časovaného) na manuální a zpět. Při manuálním krokování je nutné pro každý následující krok stisknout tlačítko číslo 2 (krokování).
6. V případě aktivovaného automatického krokování umožňuje změnu rychlosti jednotlivých kroků v intervalu od 100ms – 1000ms. Aktuální stav se zobrazuje nad posuvník

2.2 Pohyb želvy a práce s plátnem

Příkaz	**Alternativa**	**Popis**	**Příklad**
forward <x>	**fd <x>**	Přesunutí želvy o x bodů vpřed.	*forward 10*
back <x>	**bk<x>**	Přesunutí želvy o x bodů zpět.	*back 10*
left <x>	**lt <x>**	Natočení želvy o x stupňů vlevo.	*left 90*
right <x>	**rt <x>**	Natočení želvy o x stupňů vpravo.	*right 90*
penup	**pu**	Nadzvednutí želvy, želva přestane kreslit.	*penup*
pendown	**pd**	Položení želvy, želva začne kreslit.	*pendown*
showturtle	**st**	Zobrazení želvy, želva bude viditelná.	*showturtle*
hideturtle	**ht**	Skrytí želvy, želva bude neviditelná.	*hideturtle*
home		Želva se postaví doprostřed plátna (do své výchozí pozice) .	*home*
clean		Smaže celé plátno.	*clean*
clearscreen	**cs**	Provede příkaz home a clean.	*clearscreen*

2.3 Řídící struktury

Příkaz	Popis výrazu	Příklad	
repeat	Želva zopakuje blok příkazů. Následuje číslo určující počet opakování a blok kódu, který se má opakovat. Tento kód je uzavřený v hranatých závorkách	`repeat 4 [ forward 100 right 90 ]`	
if	Jednoduché větvení. Pokud želva vyhodnotí výraz <*C*> jako pravdivý, pak provede následující blok příkazů.	`if (5 < 10) [ left 90 forward 100 ]`	
ifelse	Příkaz ifelse je podobný příkazu if, jen za blokem kódu (IB) následuje další blok (IB2), který se provede v případě, že je výraz vyhodnocen jako false (nulový).	`if 1 > 4 [ left 90 fd 50 ] [ right 90 fd 50 ]`	
to … end	Definice funkce *nazev*. Parametry funkce jsou proměnné začínající dvojtečkou a jsou odděleny mezerou. Následuje tělo funkce.	`to ctverec :a repeat 4 [ fd :a left 90 ] end ctverec 50`	

2.4 Matematické operace a funkce

Výraz	Popis výrazu	Příklad
(<x>)	Závorky. Je doporučeno precizně uzávorkovávat všechny matematické výrazy.	*(10 + ((sin 5) ^ 3))*
<x> + <y>	Součet čísel *x* a *y*.	*10 + 5*
<x> -	Rozdíl čísel *x* a *y*.	*10 - 5*

<*y*>		
<*x*> * <*y*>	Součin čísel *x* a *y*.	*10 * 5*
<*x*> / <*y*>	Podíl čísel *x* a *y*.	*10 / 5*
<*x*> ^ <*y*>	Mocnina, *x*.	*10 ^ 2*
sqrt <*x*>	Druhá odmocnina z čísla *x*.	*sqrt 100*
sin <*x*>	Goniometrická funkce sinus. Číslo *x* je úhel ve stupních.	*sin 45*
cos <*x*>	Goniometrická funkce cosinus. Číslo *x* je úhel ve stupních.	*cos 45*
tan <*x*>	Goniometrická funkce tangens. Číslo *x* je úhel ve stupních.	*tan 45*

3. Python

Python je dynamický interpretovaný jazyk. Někdy bývá zařazován mezi takzvané skriptovací jazyky. Jeho možnosti jsou ale větší. Python byl navržen tak, aby umožňoval tvorbu rozsáhlých, plnohodnotných aplikací.

K význačným vlastnostem jazyka Python patří jeho jednoduchost z hlediska učení. Bývá dokonce považován za jeden z nejvhodnějších programovacích jazyků pro začátečníky. Další vlastností jazyka Python je produktivnost z hlediska rychlosti psaní programů. Týká se to jak nejjednodušších programů, tak aplikací velmi rozsáhlých. Také se snadno vkládá do jiných aplikací (embedding), kde pak slouží jako jejich skriptovací jazyk. Tím lze aplikacím psaným v kompilovaných programovacích jazycích dodávat chybějící pružnost.

3.1 Historie

Python byl vytvořen v roce 1990-1991 Guido van Rossumem v Matematickém centru Stitching v Nizozemsku jako následník jazyka nazývaného ABC. Guido i nadále zůstává hlavním autorem, ačkoliv je zde mnoho dalších přispěvatelů. Je možná zajímavé, že první myšlenka na tento projekt napadla autora na konci roku 1989, v čase naší sametové revoluce. Druhá podivná náhoda je i jeho příjmení: Rossum. Až moc připomíná český Rozum a i Čapkovo dílo R.U.R. (Rosum's Universal Robots).

Jméno Python pochází z Monty Python's Flying Circus. Tvůrci tohoto zábavného pořadu tvrdí, že nemá žádný význam a jim akorát připadalo zábavné. Stejně tak tvůrce Pythona Guido van Rossum říká, že v době, kdy hledal jméno pro svůj nový jazyk, běžel v televizi tento seriál a že mu toto slovo připadalo vhodné. Proto snad dosud Python nemá oficiální logo, přestože na mnoha webech se objevuje had (přesněji krajta), což je jeden z anglických významů slova Python. Přesto oba tvůrci souvislosti s hady odmítají.

3.2 Vlastnosti

Python je interpretovaný, interaktivní a objektově orientovaný programovací jazyk. Často je srovnáván s Tcl, Perl, Scheme nebo Javou.

Python se jednoduše učí a je to mocný programovací jazyk. Má výkonné vysokoúrovňové datové struktury a jednoduchý, přesto mocný, přístup k objektovému programování. Pythonovská elegantní syntaxe a dynamické typování, společně s jeho interpretovanou povahou, ho činí ideálním jazykem pro skriptování a rychlý vývoj aplikací v mnoha oblastech na většině platforem.

Pythonovský interpret a rozsáhlá standardní knihovna jsou zcela volně šiřitelné ve zdrojové nebo binární formě na všech hlavních platformách z pythonovské webové stránky Python.org. Na stejné stránce naleznete distribuce nebo odkazy na mnoho volně šiřitelných pythonovských modulů, programů a nástrojů třetích stran a i další dokumentaci.

Pythonovský interpret se dá snadno rozšířit pomocí nových funkcí a datových typů implementovaných v C nebo C++ (nebo jiného jazyka volatelného z C). Je také vhodný jako rozšiřující jazyk pro aplikace, které si uživatel přizpůsobuje svým potřebám.

3.3 Verze

16. 10. 2000 – verze 2.0 (komunitní vývoj, full garbage collector, prvotní podpora pro Unicode...)

3. 12. 2008 – verze 3.0, záměrně první zpětně nekompatibilní (skutečná podpora Unicode'u, sjednocení syntaxe a „vnitřností“, vyčištění systémové knihovny...)

4. Práce s Pythonem

4.1 Pohyb želvy

forward( )	fd( )	Posun želvy o x kroků dopředu	fd(25)
back( ), backward( )	bk( )	Posun želvy o x kroků dozadu	bk(25)
right( )	rt( )	Otočení želvy o x stupňů doprava	rt(90)
left( )	lt( )	Otočení želvy o x stupňů doleva	lt(90)
goto( ), setposition( )	setpos( )	Přesunutí želvy do určité pozice o souřadnicích x, y. V případě že je pero dole, nakreslí želva čáru.	goto(60,30)
setx( )		Nastavení souřadnice x	setx(10)
sety( )		Nastavení souřadnice y	sety(10)
setheading( )	seth( )	Nastavení směru želvy ve stupních: 0 - východ	seth(90)

		90- sever 180 - západ 270 - jih	
home( )		Želva se postaví doprostřed plátna (do své výchozí pozice) .	home( )
circle( )		Želva vytvoří kruh	circle(50)
dot( )		Želva vytvoří puntík	dot( )
undo( )		Smaže poslední krok želvy	Undo(
speed( )		Nastavení rychlosti želvy: “nejrychlejší” - 0 “rychlý” - 10 “normální” - 6 “pomalý” - 3 “nejpomalejší” - 1	sped(9) speed(“normal”)

4.2 Práce s perem

pendown( )	down( ), pd( )	Položení pera želvy - želva kreslí čáru když se pohybuje.	pd()
penup( )	up( ), pu( )	Zvednutí pera želvy - želva nekreslí čáru když se pohybuje.	pu()
pensize( ), widht( )		nastavení tloušťky čáry pera.	pensize(3)

4.3 Nastavení barvy čáry a výplně

color( )	Nastavení barvy čáry pera i výplně.	color(“yellow”, “red”)
pencolor( )	Nastavení barvy čáry pera.	pencolor(“yellow”)
fillcolor( )	Nastavení barvy výplně.	fillcolor(“red”)
bgcolor( )	Nastavení barvy celého plátna.	bgcolor(“black”)
begin_fill( )	Obraz se bude vyplňovat.	begin_fill( )
end_fill( )	Ukončení výplně.	end_fill( )

4.4 Smazání plátna

reset( )	Smazání všech obrazců nakreslených želvou a posun želvy do výchozí pozice.
clear	Smazání všech obrazců nakreslených želvou - želva se nepohne.

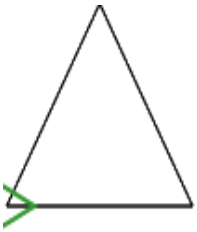
4.5 Objevení a skrytí želvy

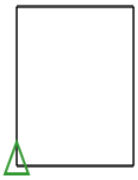
showturtle( )	st( )	Objevení želvy, želva je viditelná.
hideturtle( )	ht( )	Skrytí želvy, želva je neviditelná

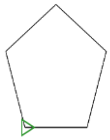
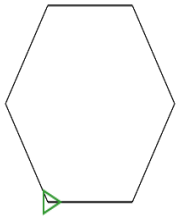
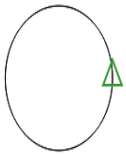
5. Příklady želví grafiky

5.1 Základní tvary

Editor	Python	Obrázek

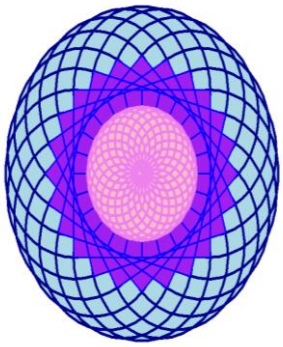
rt 30
repeat 3 [
fd 100
rt 360/3
]
	from turtle import fd, lt
	for k in range(3):
	  fd(100)
	  lt(120)
		

repeat 4 [
fd 100
lt 360/4
]
	from turtle import fd, lt
	for i in range(4):
	  fd(100)
	  lt(360/4)
		

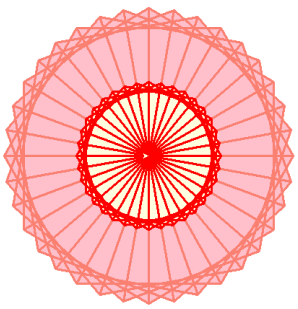
right 90 repeat 5 [forward 100 left 360/5]	from turtle import lt, fd for i in range(5): fd(100) lt(360/5)	
right 90 **repeat 6[** **forward 100** **left 360/6** **]**	from turtle import lt, fod for i in range(6): fd(100) lt(360/6)	
repeat 360[**forward 1** **left 1** **]**	from turtle import circle circle(50)	

5.2 Složitější tvary v Pythonu

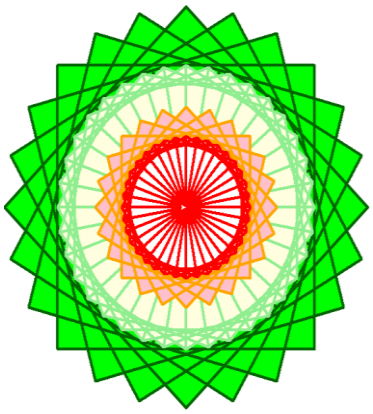
```
from turtle import *

speed(0)
pensize(5)
color("dark blue", "light blue")

begin_fill()
for k in range(24):
   circle(150)
   left(15)
end_fill()

pensize(3)
color("blue", "purple")

begin_fill()
for k in range(24):
   for k in range(4):
       forward(150)
       left(90)
   left(15)
end_fill()

pensize(5)
color("violet", "Pink")

begin_fill()
for k in range(24):
   circle(60)
   left(15)
end_fill()

done()
```



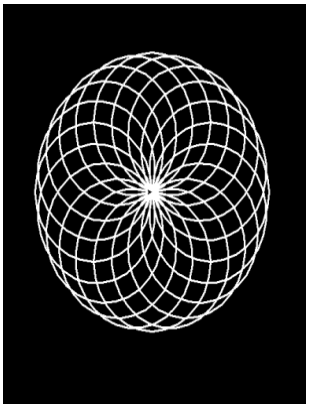
```
from turtle import *
speed(0)
pensize(3)
color("salmon", "pink")
begin_fill()
for i in range(36):
   for k in range(3):
       forward(200)
       left(360/3)
   left(10)
end_fill()

color("red", "light yellow")
begin_fill()
for i in range(36):
   for k in range(3):
       forward(150)
       left(360/3)
   left(10)
end_fill()

done()
```



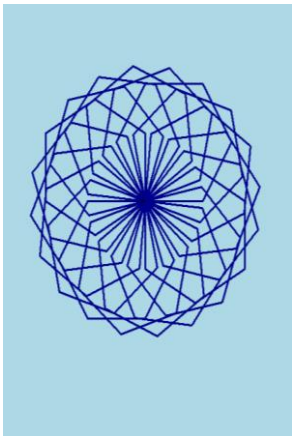
```
from turtle import *
pensize(4)
speed(0)
color("dark green", "green")
begin_fill()
for k in range(24):
   for k in range(4):
       forward(200)
       left(90)
   left(15)
end_fill()

color("light green", "light yellow")
begin_fill()
for i in range(36):
   for k in range(3):
       forward(200)
       left(360/3)
   left(10)
end_fill()

color("purple", "pink")
begin_fill()
for k in range(24):
   for k in range(4):
       forward(100)
       left(90)
   left(15)
end_fill()

color("red", "white")
begin_fill()
for i in range(36):
   for k in range(3):
       forward(100)
       left(360/3)
   left(10)
end_fill()

done()
```



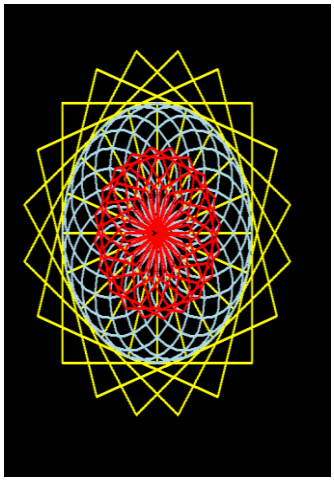
```
from turtle import *
speed(0)
bgcolor("black")
pensize(3)
pencolor("white")

for k in range(20):
   circle(100)
   left(18)

done()
```



```
from turtle import *
speed(0)
pensize(3)
bgcolor("light blue")
pencolor("dark blue")

for k in range(20):
    for i in range(6):
        forward(100)
        left(360/6)
    left(18)
done()
```

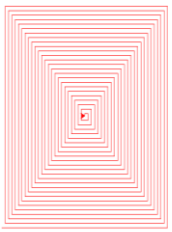


```
from turtle import *
speed(0)
pensize(3)
bgcolor("black")

pencolor("yellow")
for k in range(20):
    for i in range(4):
        forward(150)
        left(360/4)
    left(18)

pencolor("light blue")
for k in range(20):
    circle(75)
    left(18)

pencolor("red")
for k in range(20):
    for i in range(6):
        forward(50)
        left(360/6)
    left(18)

done()
```

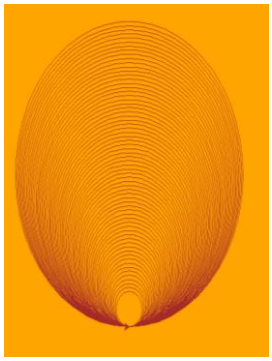


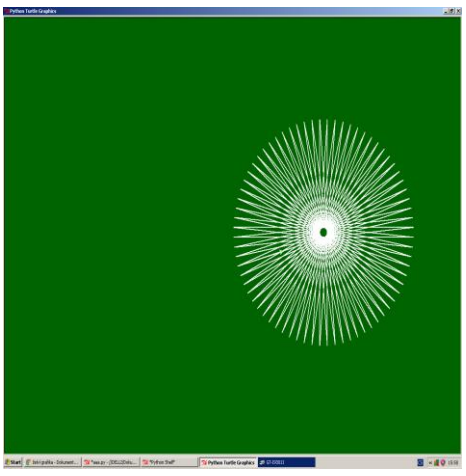
```
from turtle import *
speed(0)
bgcolor("light yellow")
color("green")

x = 1
while x < 400:
    fd(25 + x)
    rt(72)
```

`    x = x+1.5`	

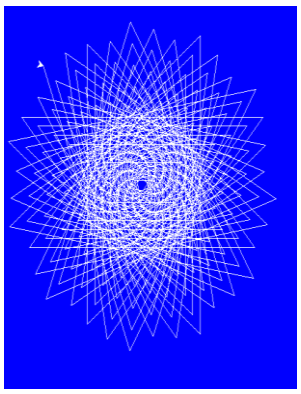
`from turtle import *`
 	
`speed(0)`
`color("red")`
 	
`side=400`
 	
`for i in range (100):`
`  fd(side)`
`  lt(90)`
`  side=side-4`
	

`from turtle import *`
 	
`speed(0)`
`color("light blue")`
 	
`side=400`
 	
`for i in range (100):`
`  fd(side)`
`  lt(120)`
`  side=side-4`	

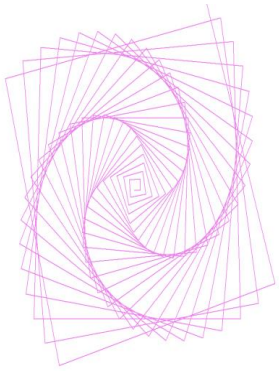
`from turtle import *`
`speed(0)`
`bgcolor("orange")`
`color("brown")`
 	
`x = 1`
`while x < 200:`
`    circle(25 + x)`
 	
`    x = x+3`
	

`from turtle import *`
`speed(0)`
`pensize(2)`
`bgcolor("dark green")`
`color("white")`
 	
`x = 1`
`while x < 400:`
`    fd(500 + x)`
`    rt(185)`
	

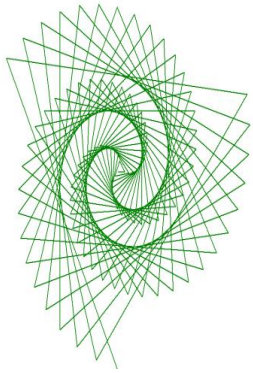
```
from turtle import *
speed(0)
bgcolor("blue")
color("white")

x = 1
while x < 400:
   fd(25 + x)
   rt(360/3)
   rt(10)

   x = x+1.5
```



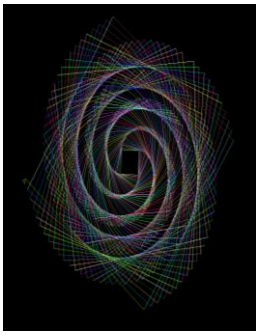
```
from turtle import *

speed(0)
pensize(2)
color("violet")

side=20

for i in range (100):
 fd(side)
 lt(92)
 side=side+7

penup()
goto(500,500)
```



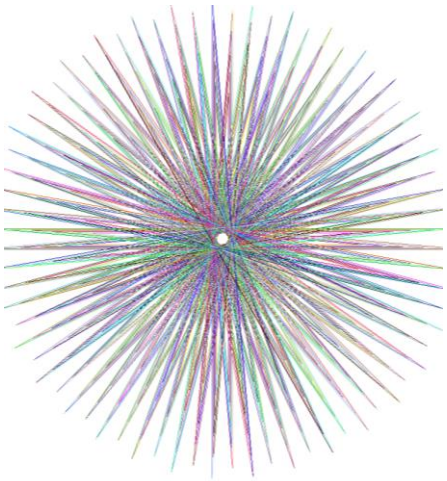
```
from turtle import *

speed(0)
pensize(2)
color("green")
side=20

for i in range (100):
 fd(side)
 lt((360/3)+3)
 side=side+7

penup()
goto(500,500)
```



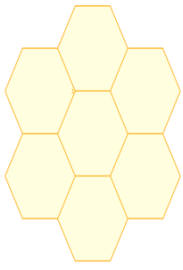
```
from turtle import *
from random import randint

speed(0)
bgcolor('black')

x = 1
while x < 400:

   r = randint(0,255)
   g = randint(0,255)
   b = randint(0,255)
   colormode(255)
   pencolor(r,g,b)

   fd(50 + x)
   rt(91)

   x = x+1
```



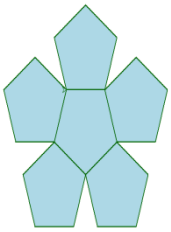
```
from turtle import *
from random import randint

speed(0)

x = 1
while x < 400:

   r = randint(0,255)
   g = randint(0,255)
   b = randint(0,255)
   colormode(255)
   pencolor(r,g,b)
   fd(450+x)
   lt(185)

   x = x+1
```



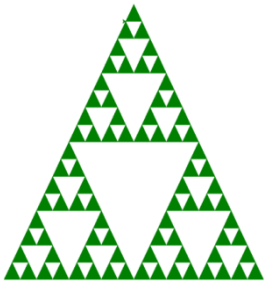
```
# žluté šestitiúhelníky (fexagony)
from turtle import *

pensize(2)
color("orange", "light yellow")

begin_fill()
def hexagon():
 for i in range(6):
     forward(100)
     left(60)

for i in range (6):
   hexagon()
   forward(100)
   right(60)
end_fill()

done()
```



```
# modré pětiúhelníky (pentagony)
from turtle import *
pensize(2)
color("dark green", "light blue")

begin_fill()
def pentagon():
  for i in range(5):
     fd(100)
     lt(360/5)

for i in range (5):
  pentagon()
  forward(100)
  right(72)
end_fill()

done()
```



```
from turtle import *

color("green")
size=800
min=64
pf=0.8660254
def S(l,x,y):
   if l>min:
       l=l/2
       S(l,x,y)
       S(l,x+l,y)
       S(l,x+l/2,y+l*pf)
   else:
       goto(x,y); pendown()
       begin_fill()
       fd(l); left(120)
       fd(l); left(120)
       fd(l)
       end_fill()
       setheading(0)
       penup(); goto(x,y)
penup()
speed(0)
S(size,-size/2,-size*pf/2.0)
done()
```



Závěr

Seminární práce pojednávala o Želví grafice v Pythonu a výukovém webovém editoru studenta Masarykovy univerzity Martina Korbela. Úvodní část byla věnována historii, zároveň bylo v této části uvedeno a vysvětleno množství pojmů. Následující kapitola je zaměřena na metodiku výuky programování s množstvím názorných příkladů od nejjednodušších po složitější.

Literatura

http://programujte.com/clanek/1970010106-python-popis-jazyka/

https://odevzdej.cz/auth/podob/2c57f8e8d9321d16/b73ef336a18bb122/

http://cs.wikipedia.org/wiki/Logo_(programovac%C3%AD_jazyk)

http://cs.wikipedia.org/wiki/Logo_programming_language

https://cs.m.wikipedia.org/wiki/Python_(programovac%C3%AD_jazyk)