IB111

Základy programování

František Lachman lachmanfrantisek@mail.muni.cz

cvičení 4

9. říjen 2018

Osnova

- kontrolní otázky
- (pseudo)náhodná čísla
- simulace a nalýzy
- (debugging)
- první domácí úkol
- zadání druhé domácí úlohy

Kontrolní otázky

Docházka

Jakým způsobem vygenerujeme náhodné číslo od `1` do `20`?

```
import random
random.randint(1,6)
```

Jakým způsovem vygenerujeme náhodné číslo v intervalu `[0,1)`?

```
import random
random.random()
```

Jak generovat vždy stejnou posloupnost "náhodných" čísel?

```
random.seed(42)
print(random.random())

random.seed(42)
print(random.random())
```

Je u datové struktury seznam důležité pořadí prvků?

Může v Pythonu seznam obsahovat položky různého typu?

```
["hello", 3]
```

Jakým příkazem přidáme do seznamu nový prvek?

```
my_list = [4,5]
print(my_list)
my_list.append(6)
print(my_list)
```

Co znamená „indexování od nuly“?

Co znamená zápis alist[3:7]?

```
alist[3:7]
```

Proč není dobrý nápad dát proměnné obsahující seznam jméno list?

Řetězec je v mnoha ohledech podobný jako „seznam znaků“. V čem se liší?

```
my_list = ['a', 'b']
my_list.append('c')
my_list[1] = 'c'
print(my_list)
```

```
my_string = "ab"
my_string[1] = 'c' # error
```

```
my_string = my_string + 'c'
print(my_string)
```

Jaký je význam funkcí chr a ord?

```
print(ord('a'))
print(chr(61))
```

Jak zjistíme délku seznamu?

```
print(len([]))
print(len(['a', 'b']))
print(len("abcd"))
```

Jak vytvořit seznam obsahující čísla od 1 do 5? (uveďte několik různých způsobů)

```
pass
```

Jak zjistíme poslední prvek seznamu? Zkuste najít 3 různé způsoby.

```
pass
```

 Debugging (ladění kódu) 

- Breakpoint
  - IDLE (editor) -> pravé tlačítko
    -> Set Breakpoint/Clear Breakpoint
- Spuštění debuggeru
  - IDLE (shell) -> Debug -> Debugger
- Zobrazení stavu proměnných, spouštění kódu po krocích, k následujícímu Breakpointu.

4.1. Hrátky s náhodností

- 4.1.1. Šestiboká kostka
  Napište funkci `dice`, která bude vracet nově vygenerované náhodné číslo simulující šestistěnnou kostku s čísly `1`-`6`.

```
def dice():
    return random.randint(1,6)
```

- ### 4.1.2. Dokud padá sudé číslo
  Napište funkci `turn`, která bude provádět házení obyčejnou šestistěnnou kostkou tak dlouho dokud nepadne liché číslo. Poté funkce vrátí celkový součet všech hozených hodů.

```
def turn():
    throw = dice()
    result = throw
    # print("throw: " + str(throw))

    while throw % 2 == 0:
        throw = dice()
        # print("throw: " + str(throw))
        result += throw

    return result
```

4.1. Hrátky s náhodností

- 4.1.4. Frekvence kostky
  Napište funkci `dice_freq`, která provede `count` hodů obyčejnou šestibokou kostkou a následně vypíše kolikrát které číslo padlo.

Tip: pro ukládání frekvencí se mohou hodit seznamy z následující kapitoly či nějaká jiná datová struktura.

```
def dice_freq(count):
    numbers = [0,0,0,0,0,0]

    for _ in range(count):
        throw = dice()
        numbers[throw -1] += 1
```

4.2. Simulace a analýzy

- 4.2.1. Opilec na cestě domů
  Opilec je na půli cesty mezi domovem a hospodou, každý krok udělá náhodně jedním směrem. Napište funkci `drunkman_simulator`, která bude simulovat opilcův pohyb. Jejímy parametry budou vzdálenost mezi domovem a hospodou a počet kroků do opilcova usnutí (tj. maximální délka simulace). Simulace skončí buď tehdy, když opilec dojede domů nebo do hospody, případně po vyčerpání počtu kroků.

```
>>> drunkman_simulator(10, 100)

home . . . . . * . . . . pub
home . . . . * . . . . . pub
home . . . * . . . . . . pub
home . . . . * . . . . . pub
home . . . . . * . . . . pub
home . . . . * . . . . . pub
home . . . . . * . . . . pub
home . . . . . * . . . pub
home . . . . . * . . . . pub
home . . . . . * . . . pub
home . . . . . * . . pub
home . . . . . * . . pub
home . . . . . * pub
home . . . . . * . pub
home . . . . . * pub
home . . . . . . pub
Ended in the pub again!
```

```
from random import randint, random

def print_path(size, position):
    print("home", end=" ")
    for i in range(size):
        if i == position:
            print("*", end=" ")
        else:
            print(".", end=" ")
    print("pub")
```

```
def drunkman_simulator(size, steps, output=False):
    if size < 3:
        print("Path too short")
        return
    position = int((size - 1) / 2)
    if output: print_path(size, position)
    for _ in range(steps):
        direction = randint(1,2)
        if direction == 1:
            position += 1
        else:
            position -= 1
        if output: print_path(size, position)
        if position == size - 1:
            if output: print("Ended in the pub again!")
            return False
        elif position == 0:
            if output: print("Got home!")
            return True
```

4.2. Simulace a analýzy

- 4.2.2. Analýza opilce
  Nejprve upravte funkci `drunkman_simulator` z předchozí příkladu tak, aby nevypisovala stav opilce (například přidáním volitelného parametru `output` a zapodmínkováním výpisu) a aby vracela `True` dojde-li opilec domů a `False` pokud ne. Následně napište funkci `drunkman_analysis`, která provede simulaci opilce `count` krát a vypíše procentuální úspěšnost dojití domů.

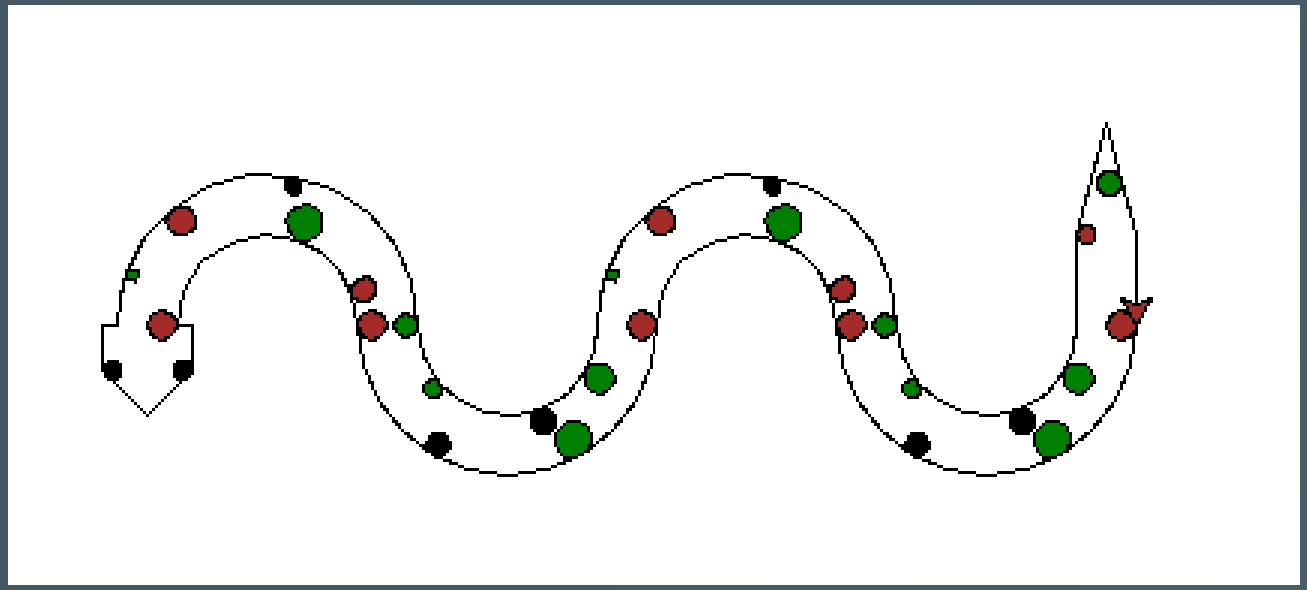
```
def drunkman_analysis(size, steps, count):
    times_home = 0
    for _ in range(count):
        if drunkman_simulator(size, steps):
            times_home += 1
    print("Arriving home in", float(times_home) * 100/coun
```

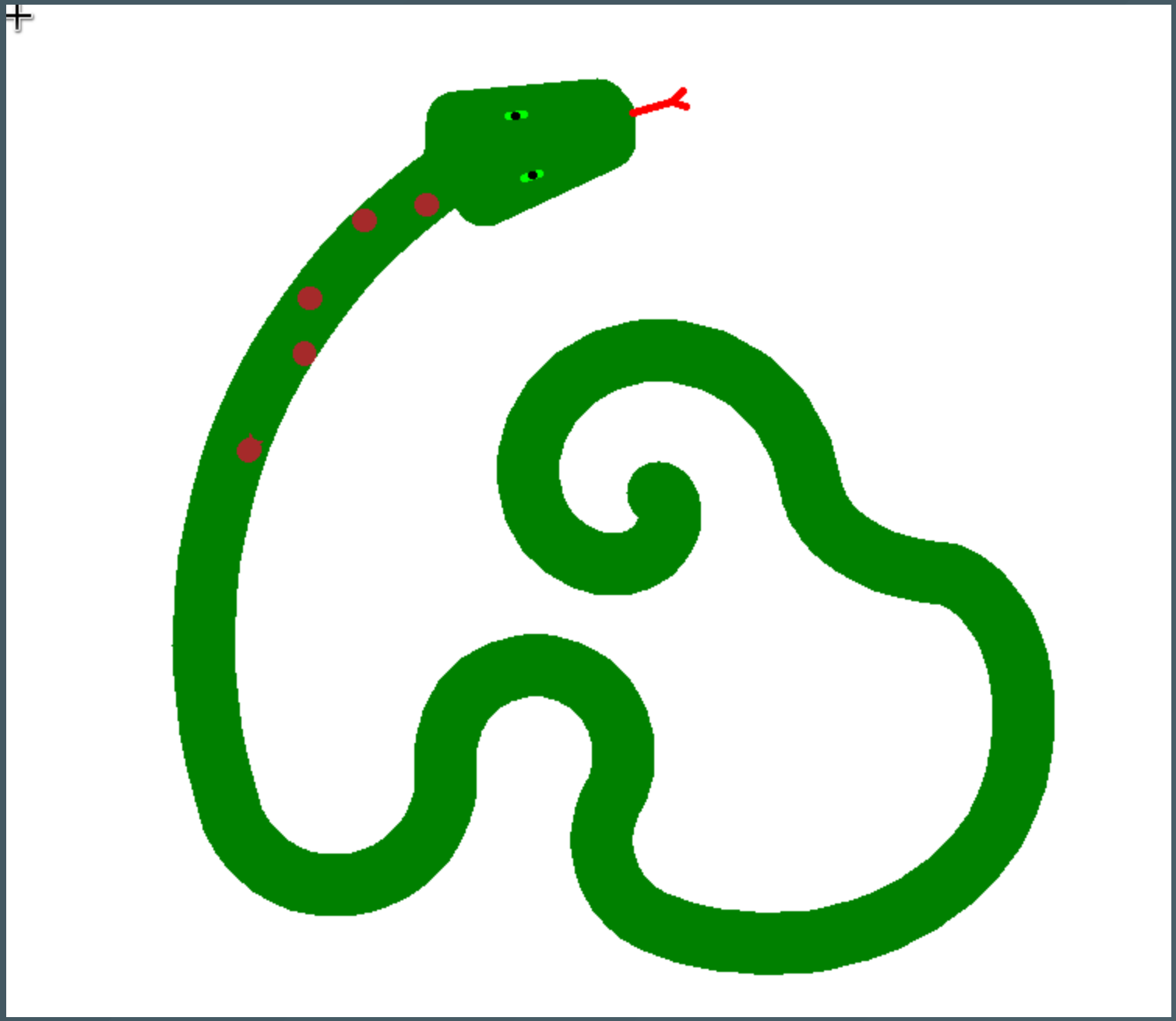
4.2. Simulace a analýzy

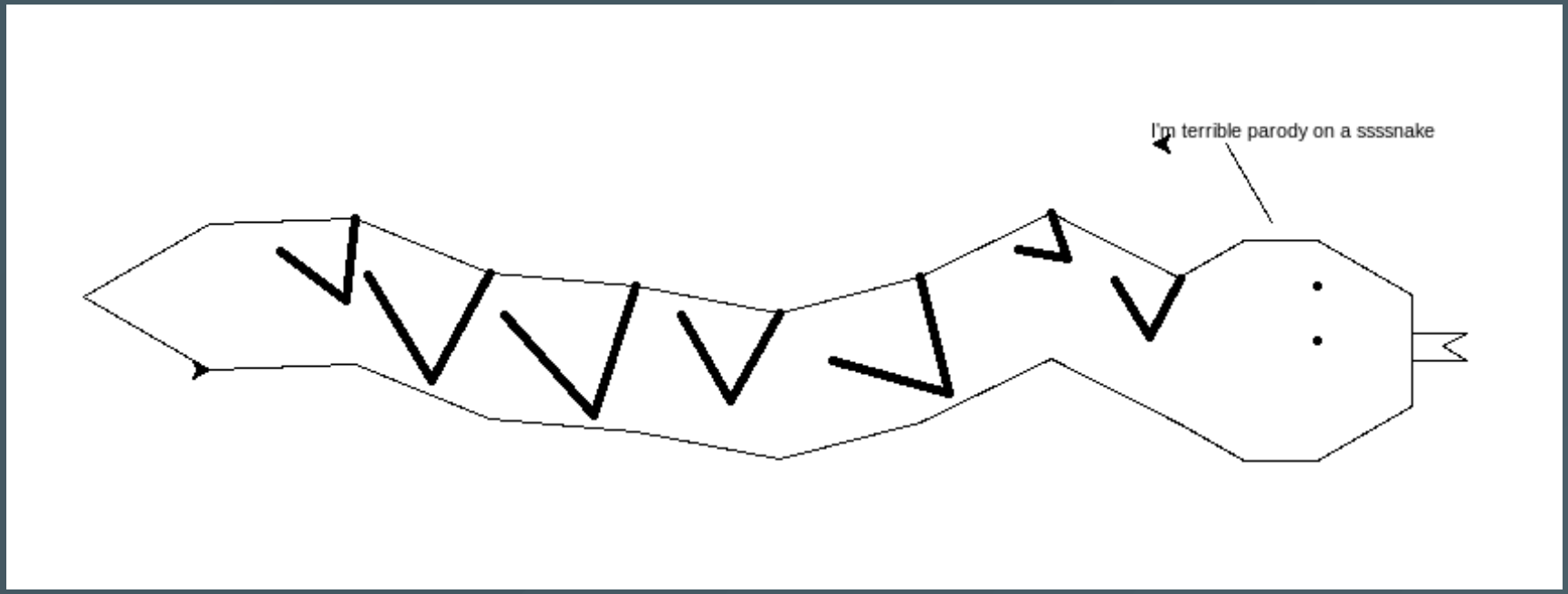
- 4.2.6. Tipující student
  Napište funkci `random_student`, která experimentálně zjistí (pro `count` pokusů) jaká je pravděpodobnost, že student natipuje alespoň polovinu `n` otázkového testu, ve kterém každá otázka má právě 4 možnosti a právě jedna z nich je správně.

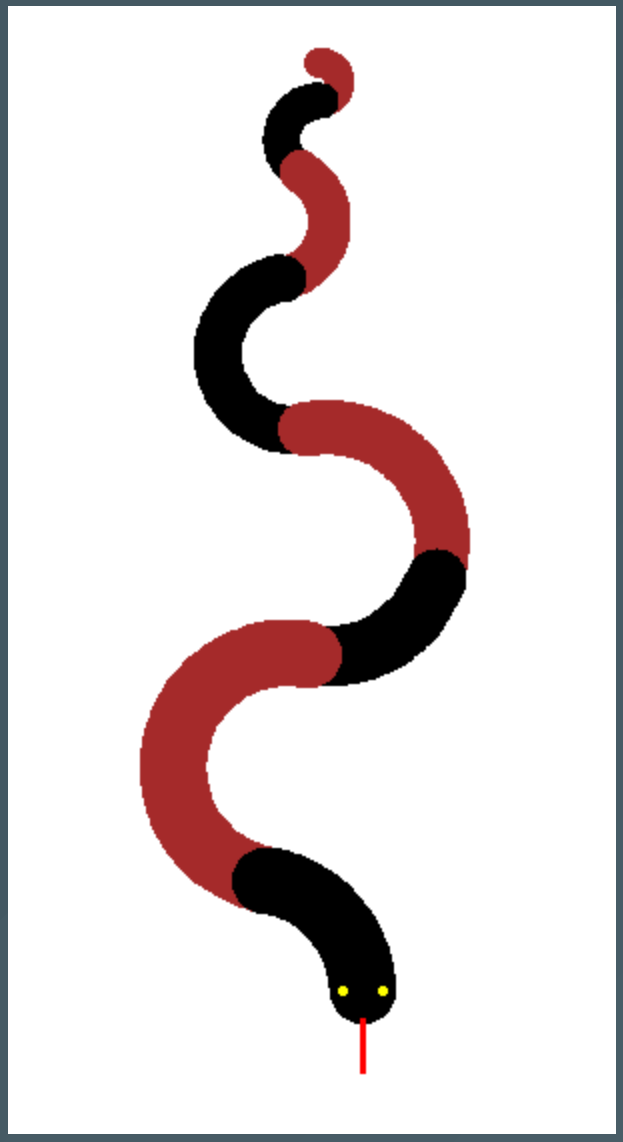
```
def random_student(n, count):
    correct_tests = 0
    for _ in range(count):
        correct_questions = 0
        for _ in range(n): # Go through test
            correct = randint(1,4)
            chosen = randint(1,4)
            if correct == chosen:
                correct_questions += 1
        if correct_questions > n/2:
            correct_tests += 1
    print("Probability of quessing at least 50 % of test i
```

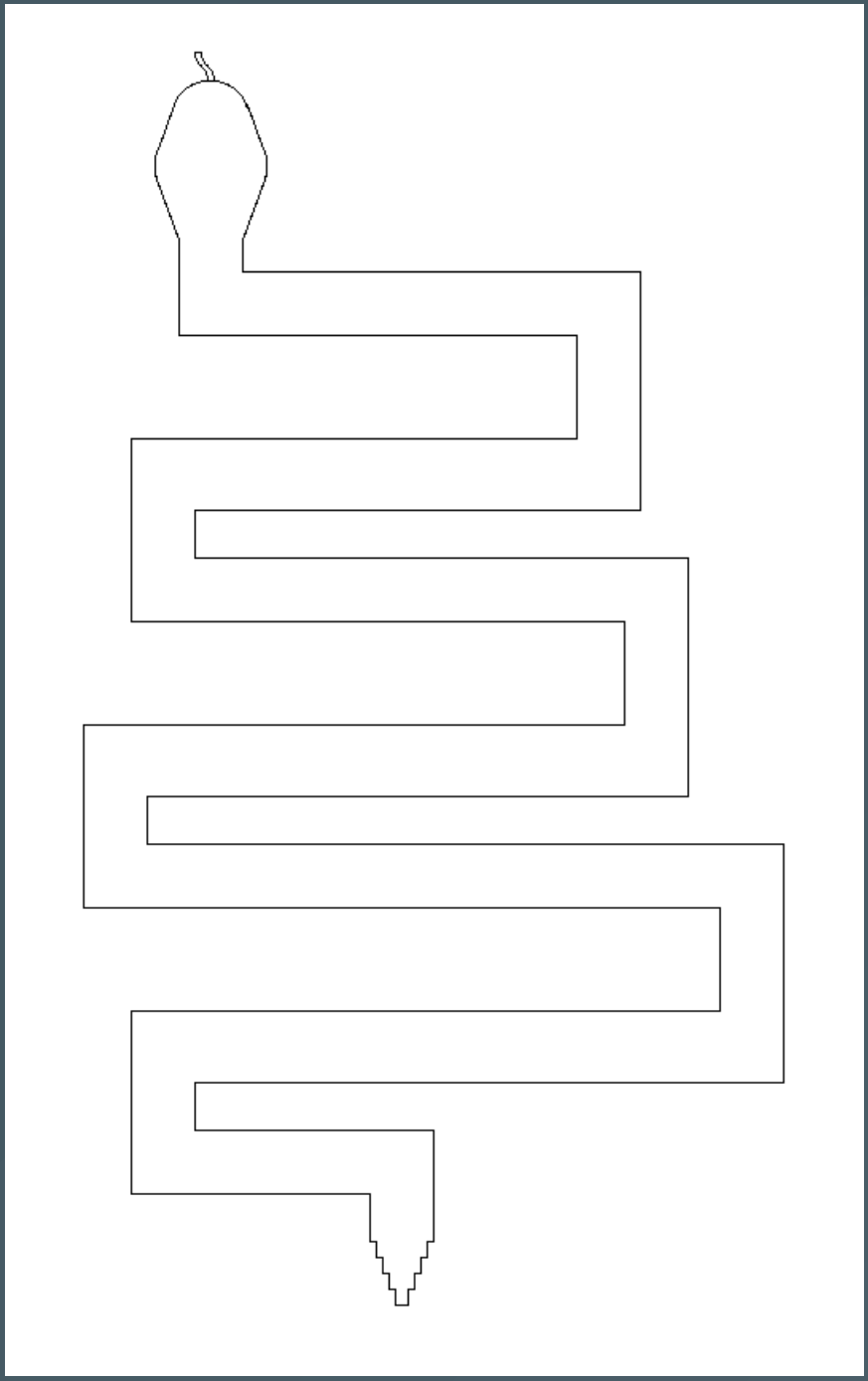
Galerie hadů

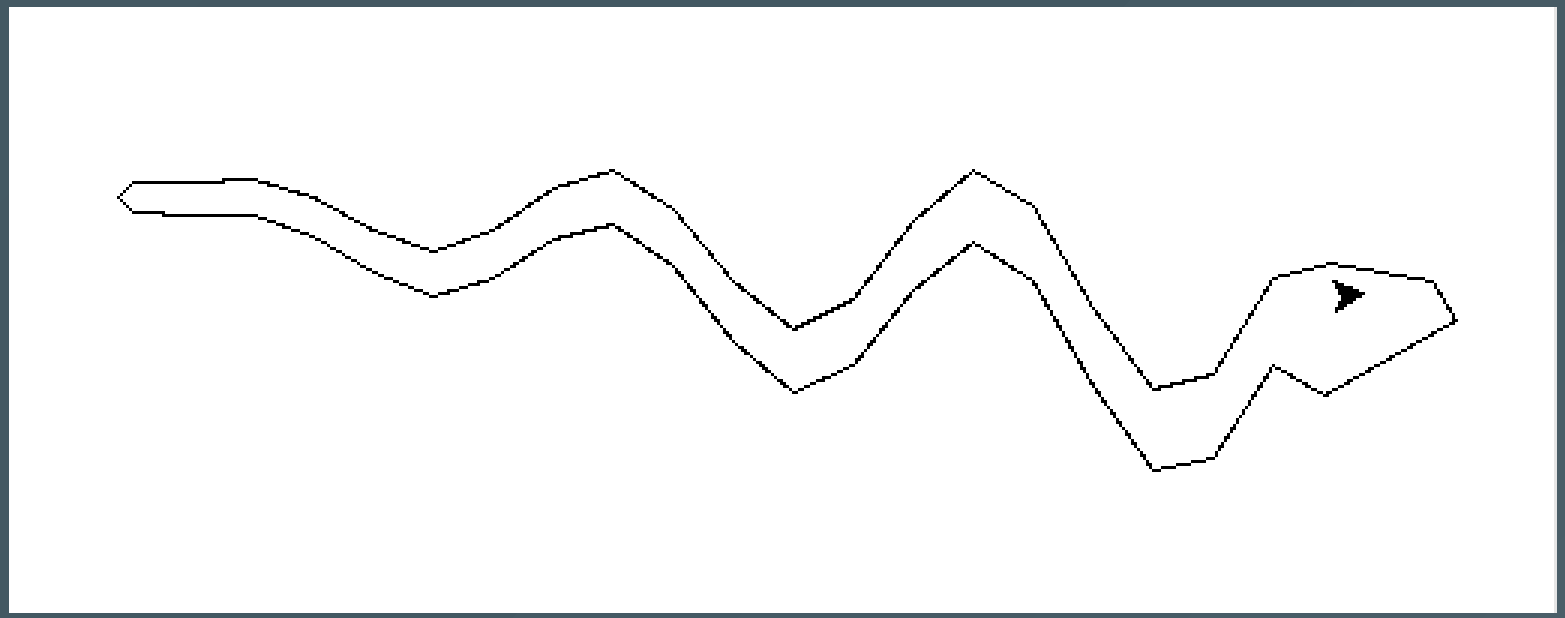


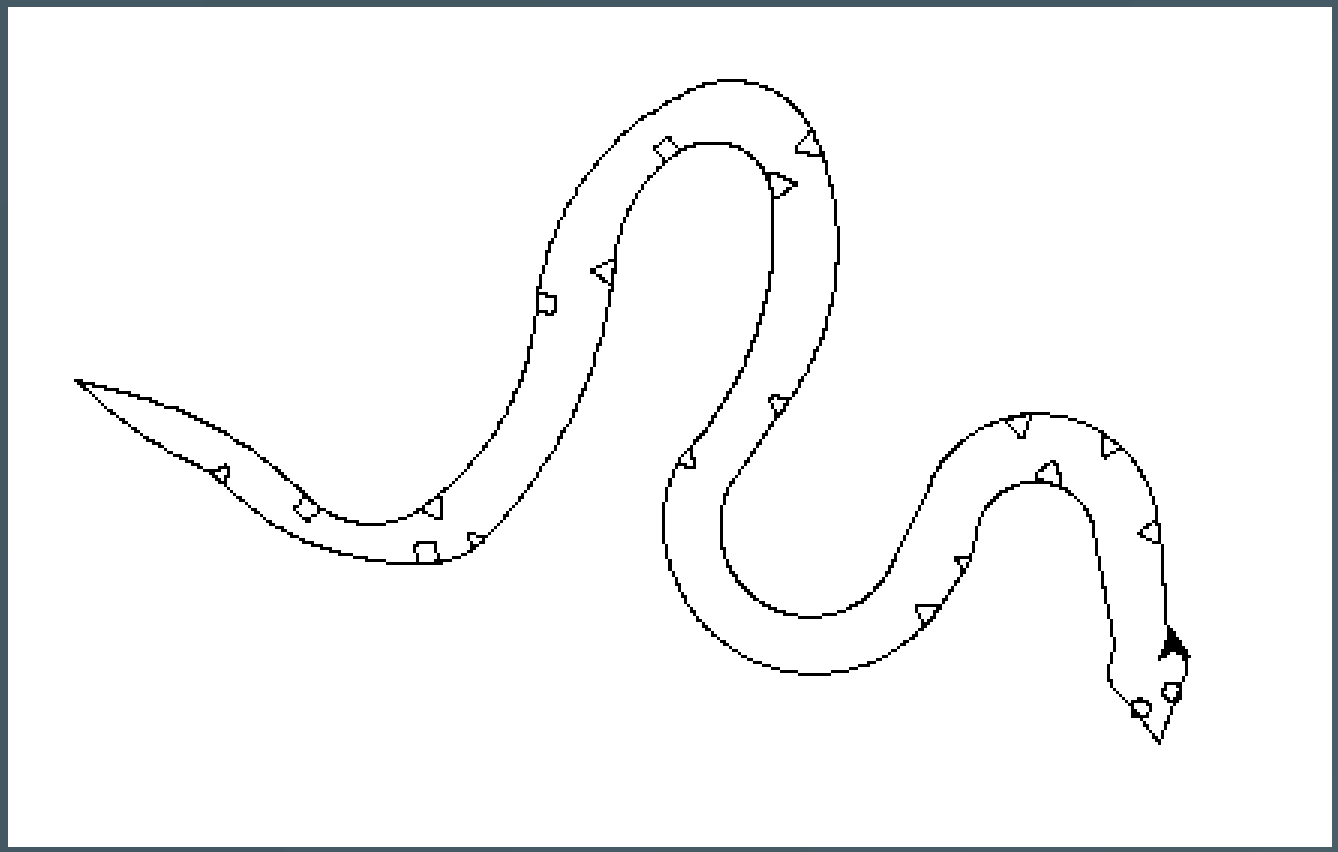


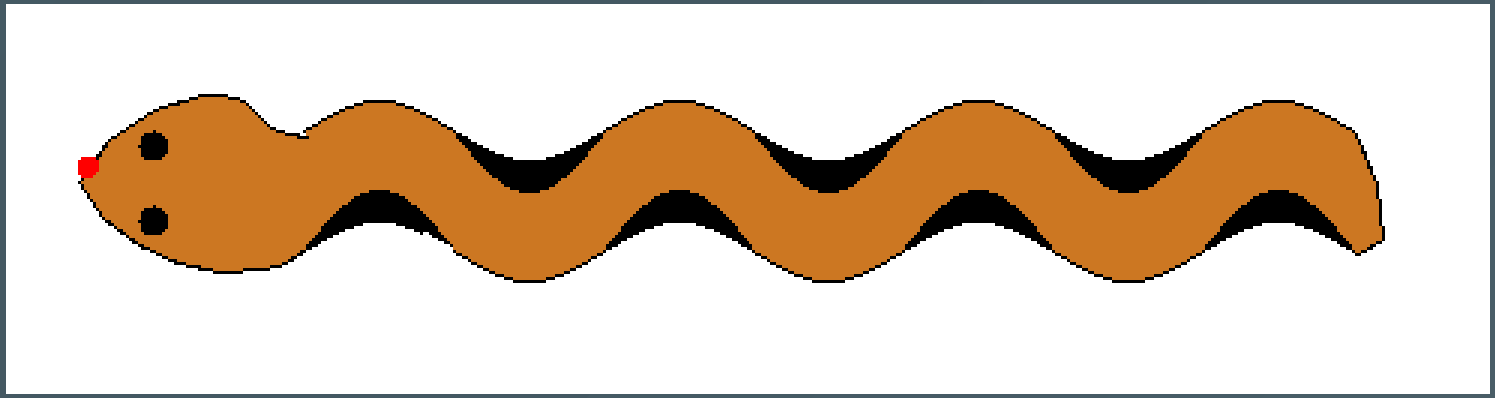

I'm terrible parody on a ssssnake

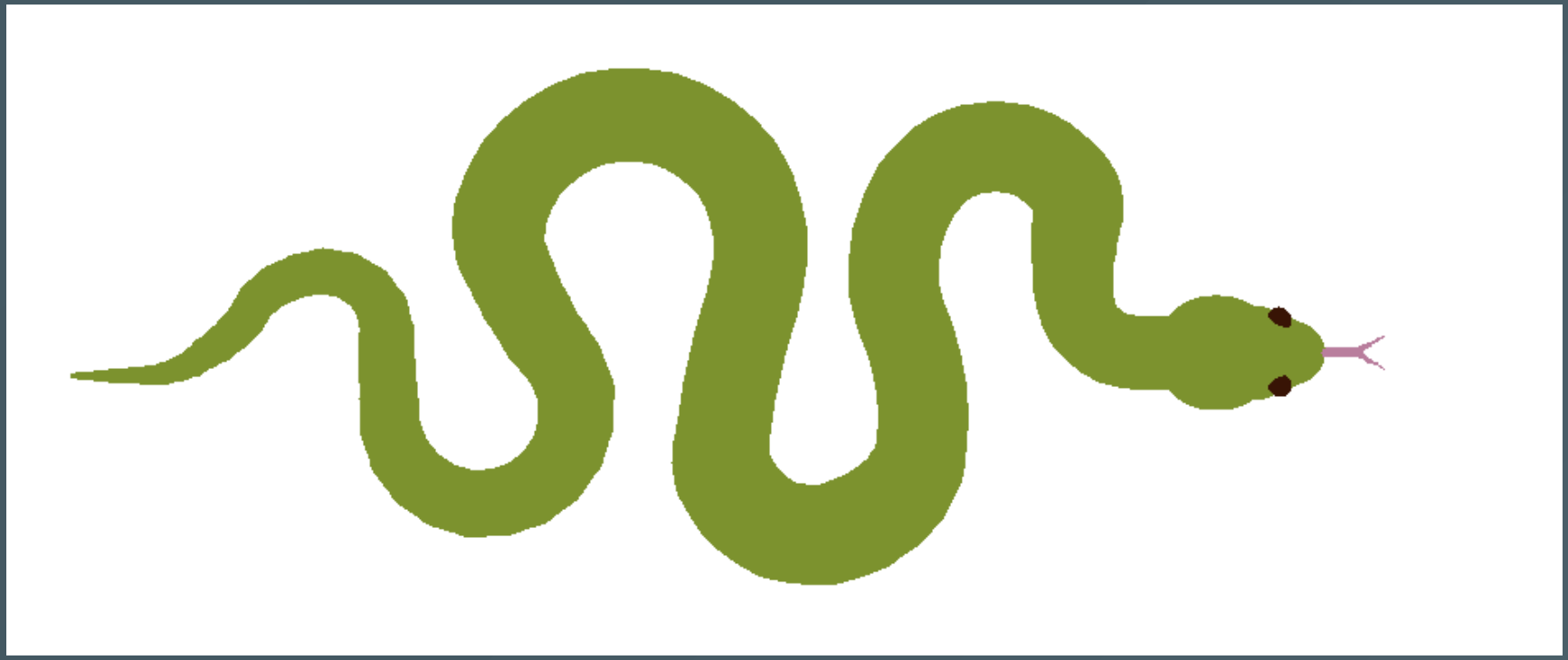


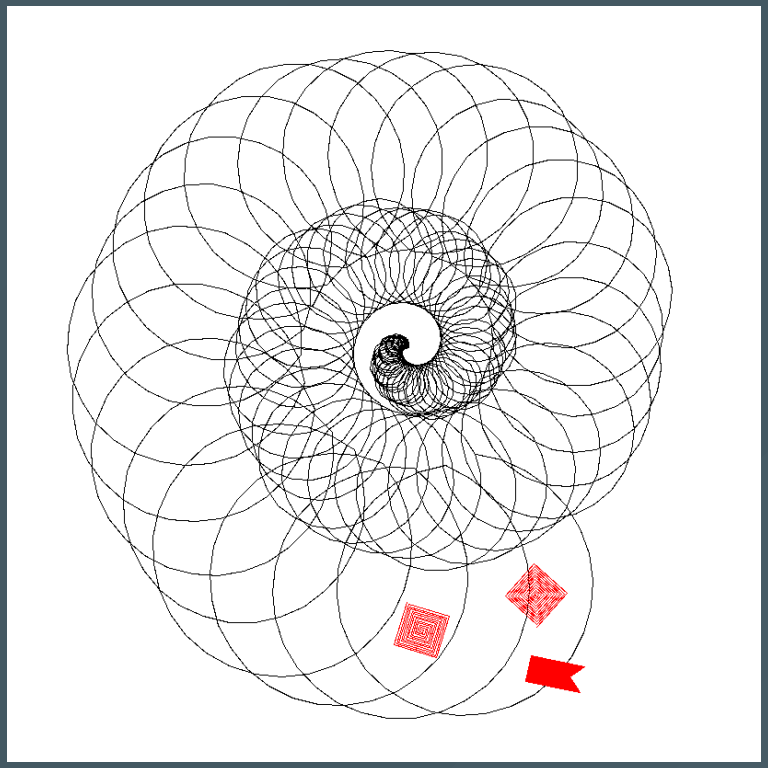


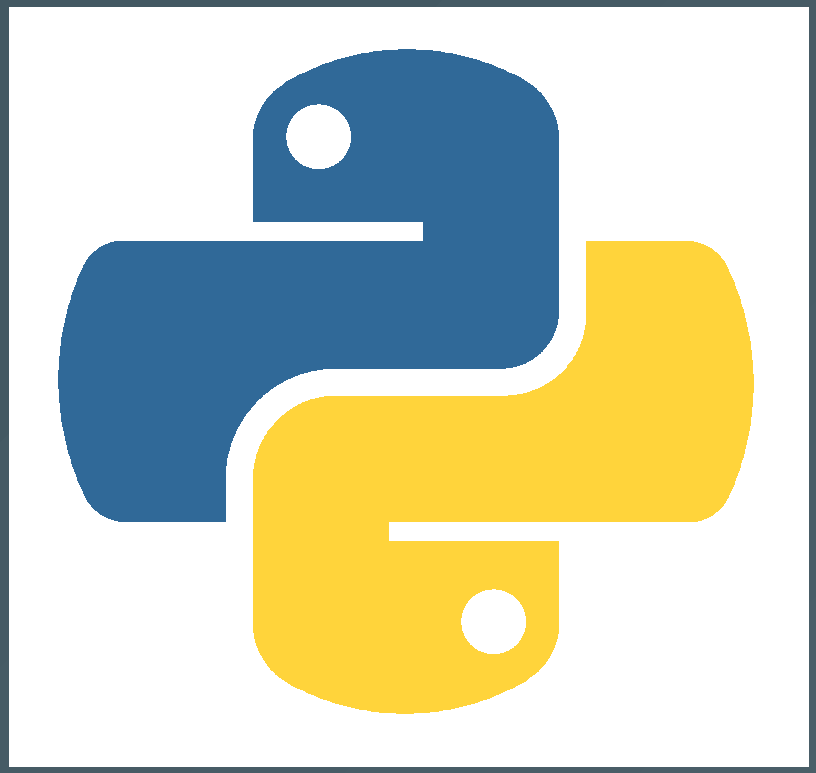


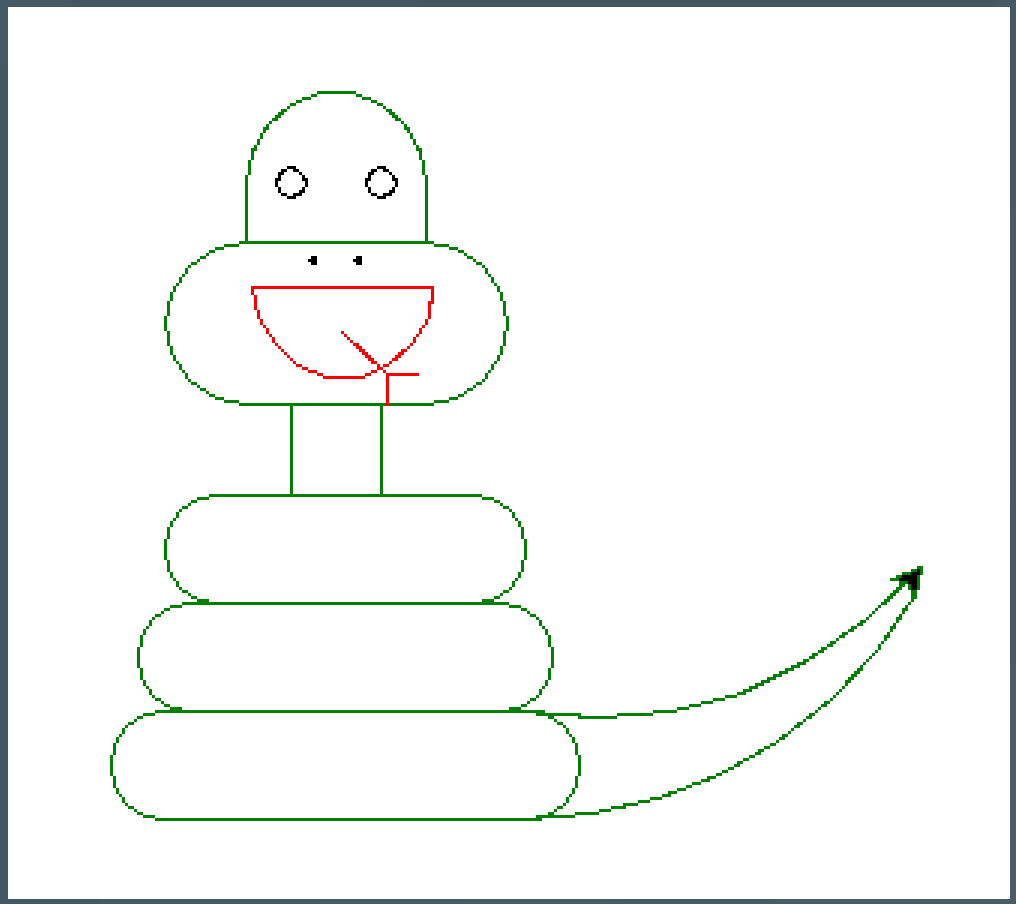


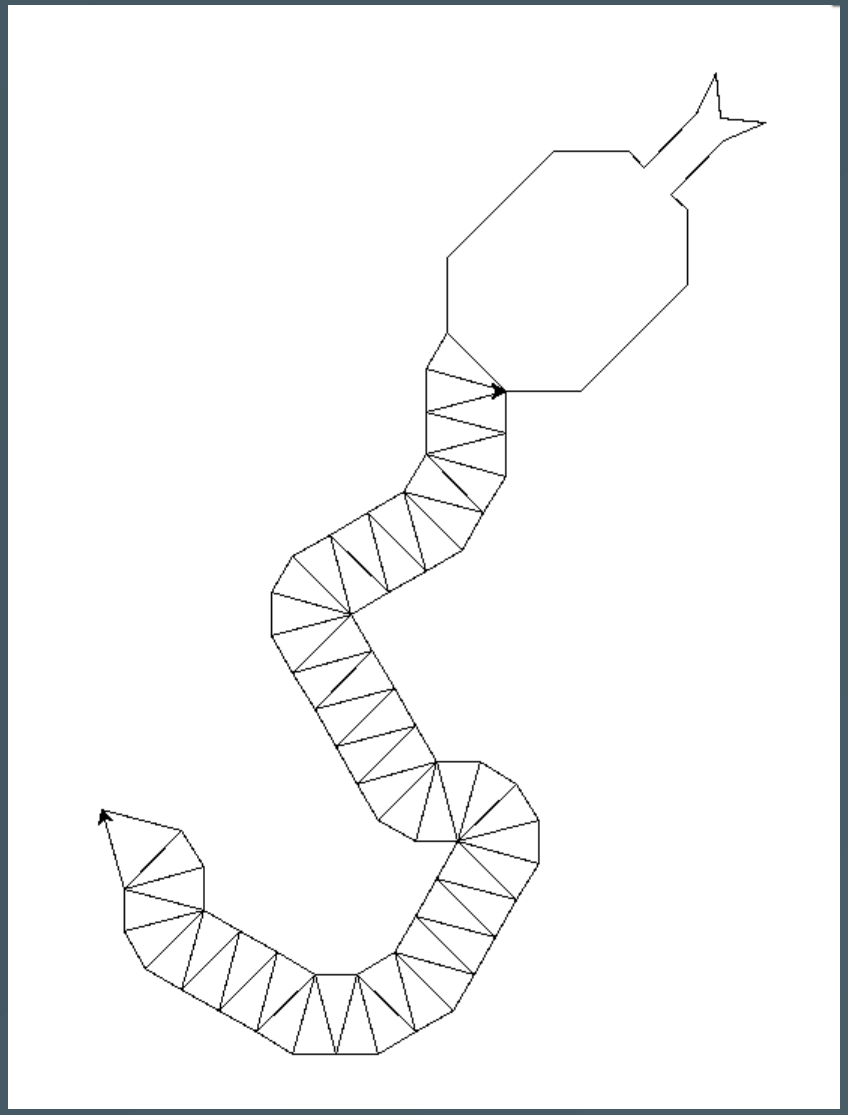


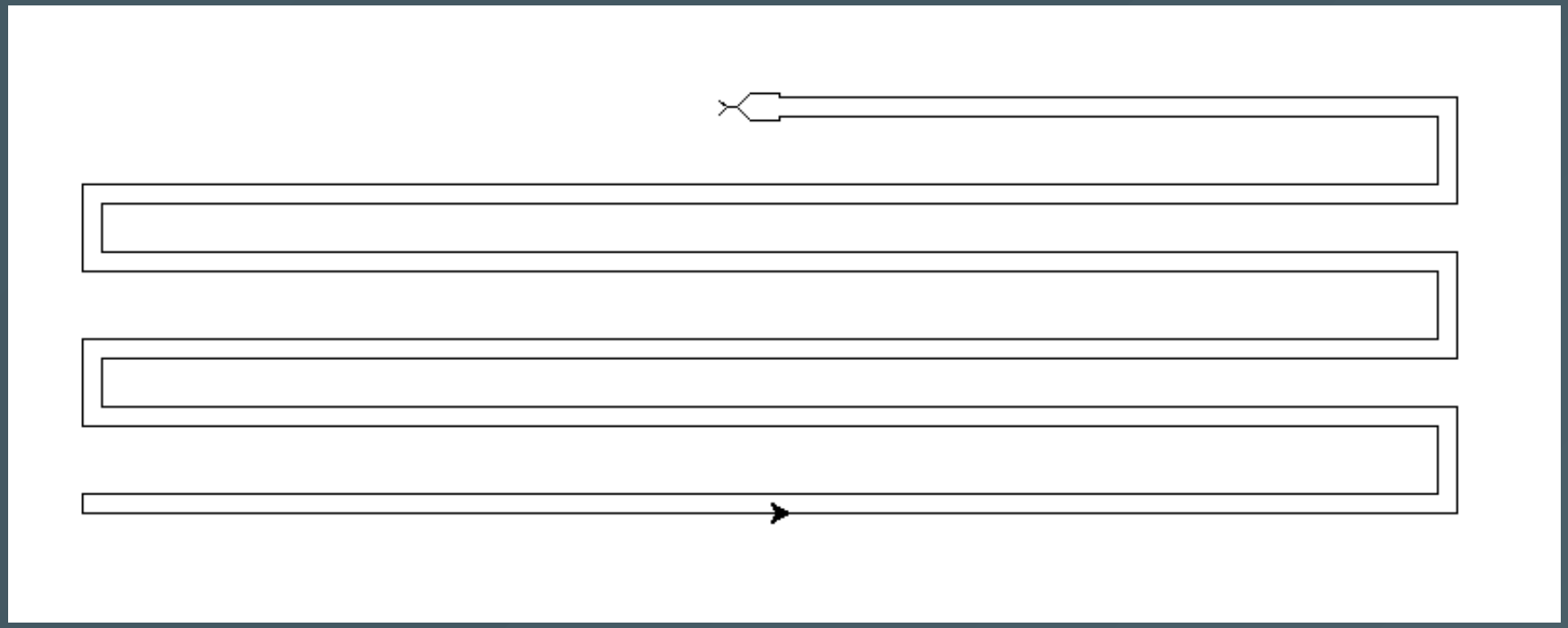




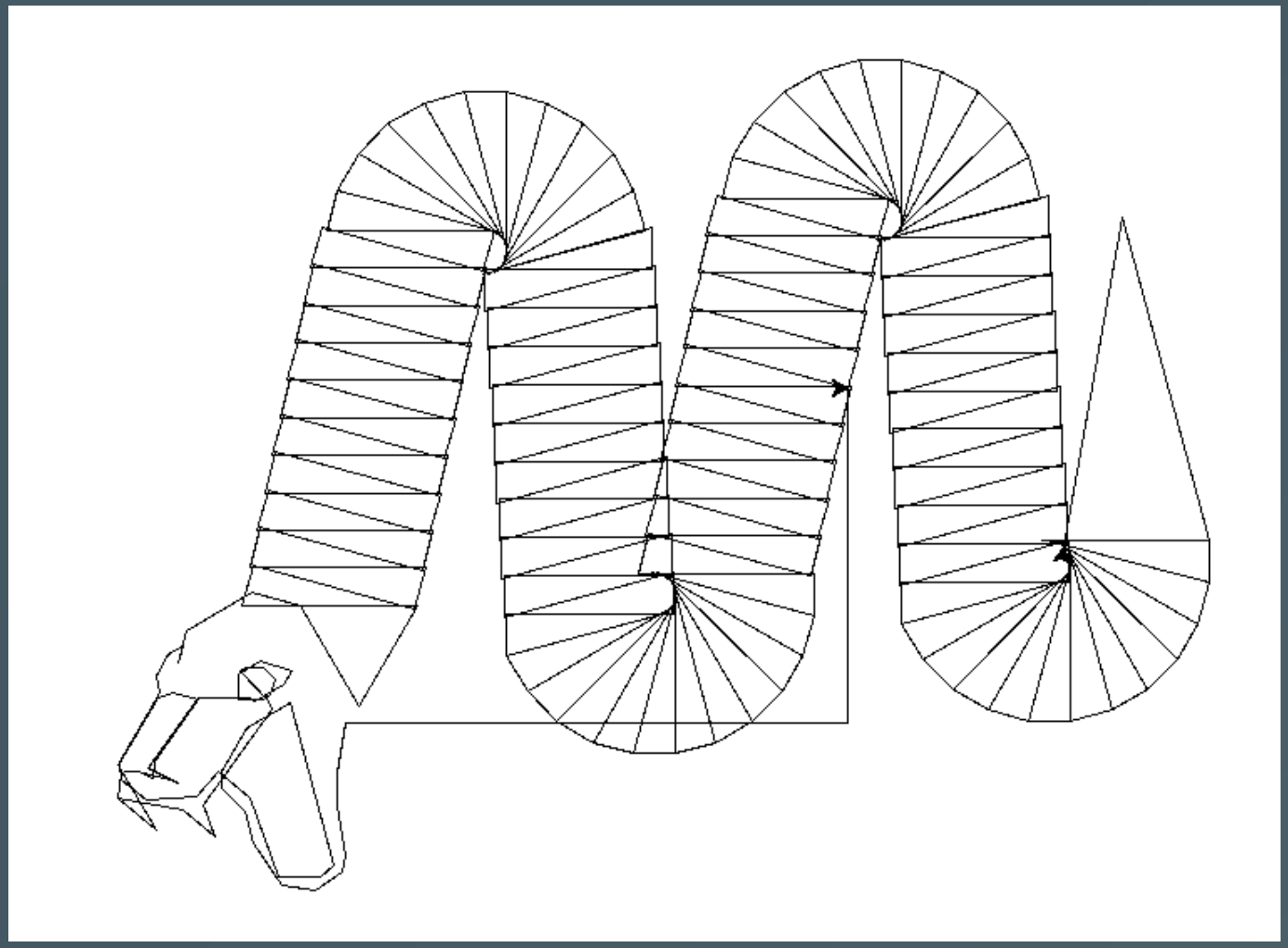


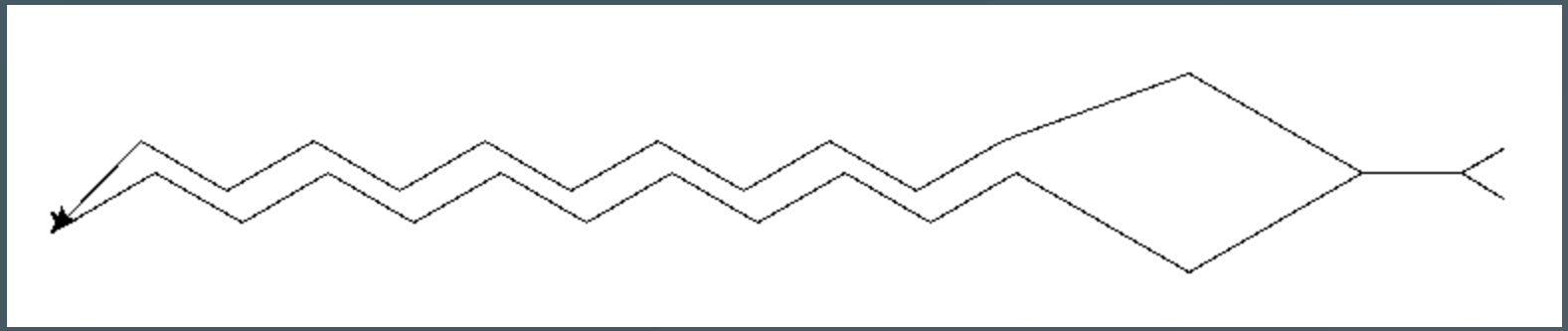


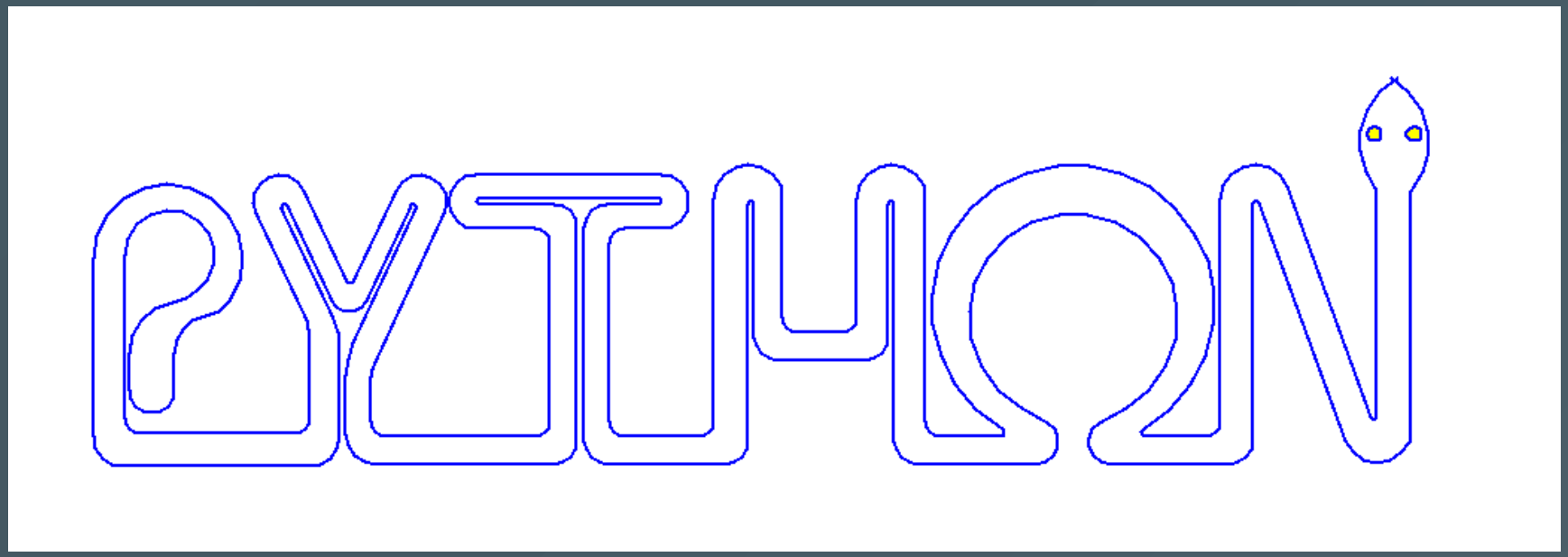


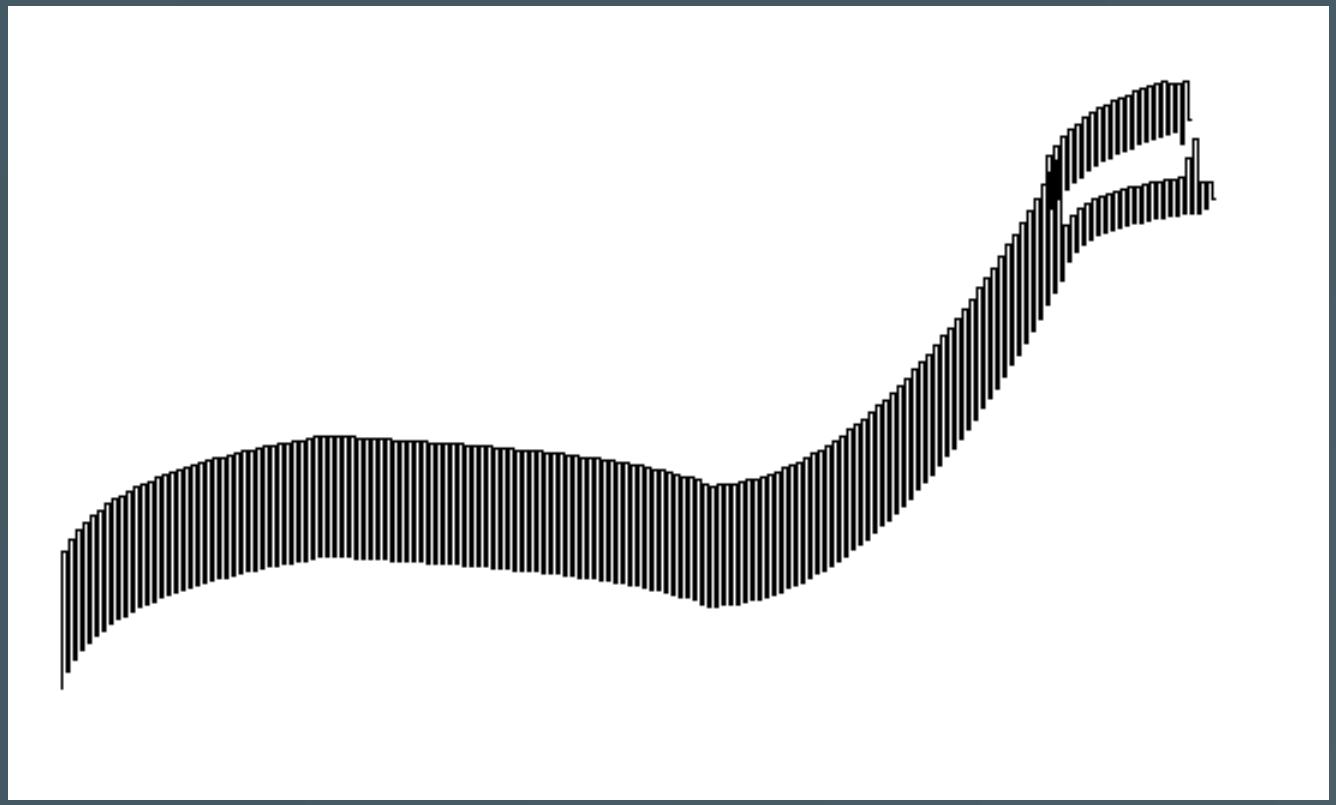


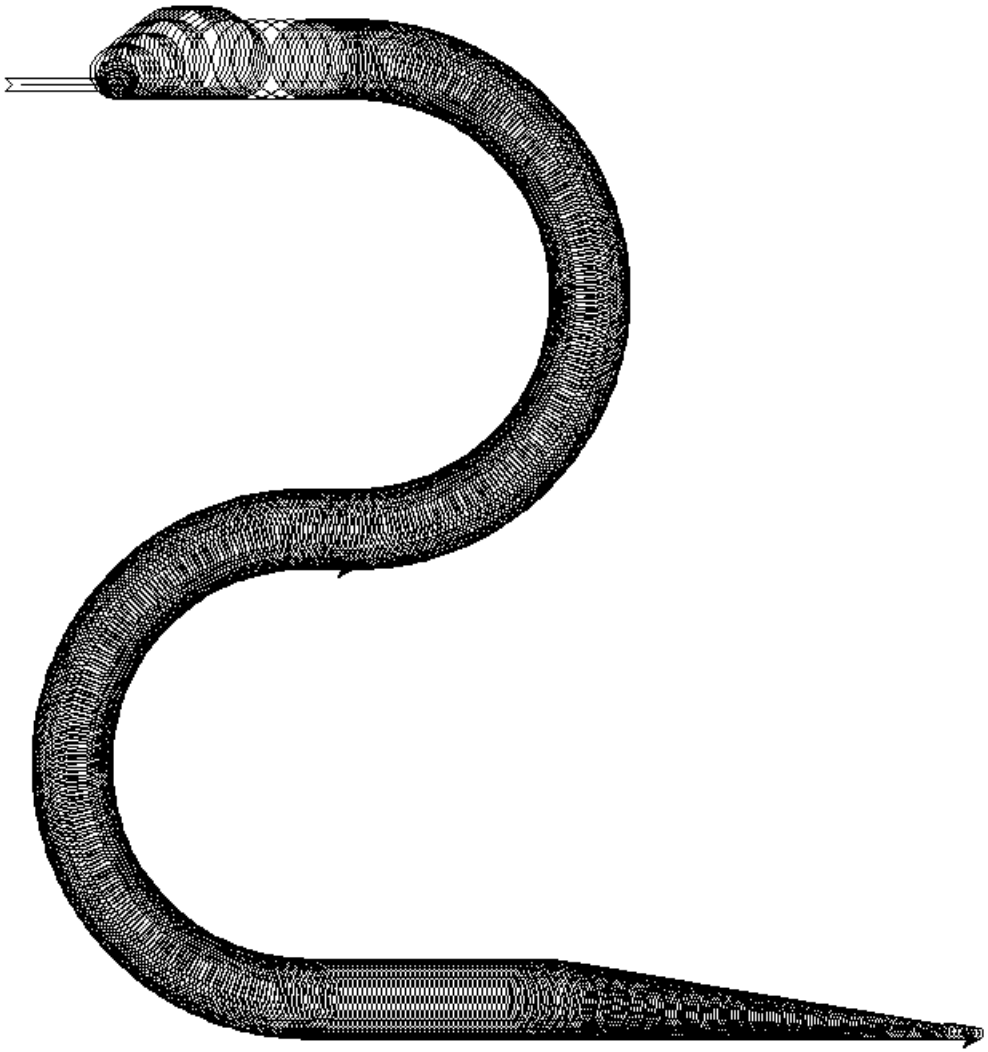
2017

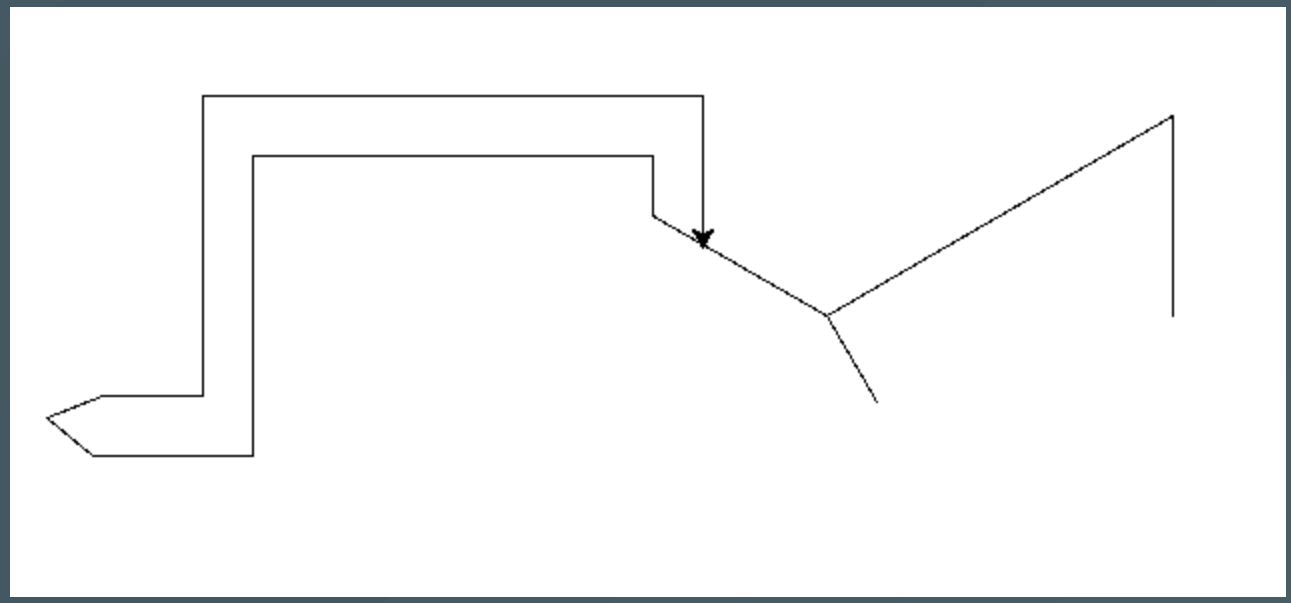


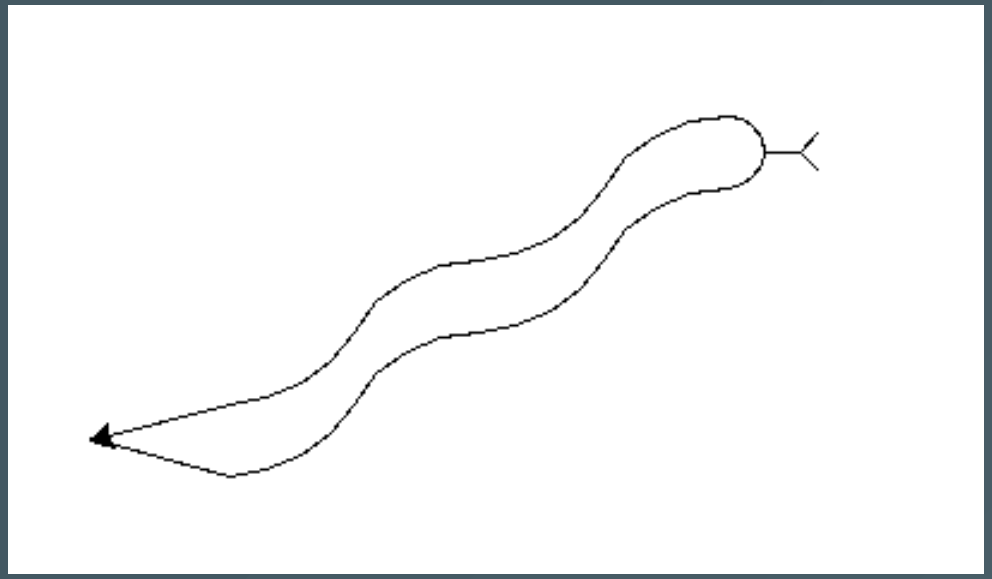


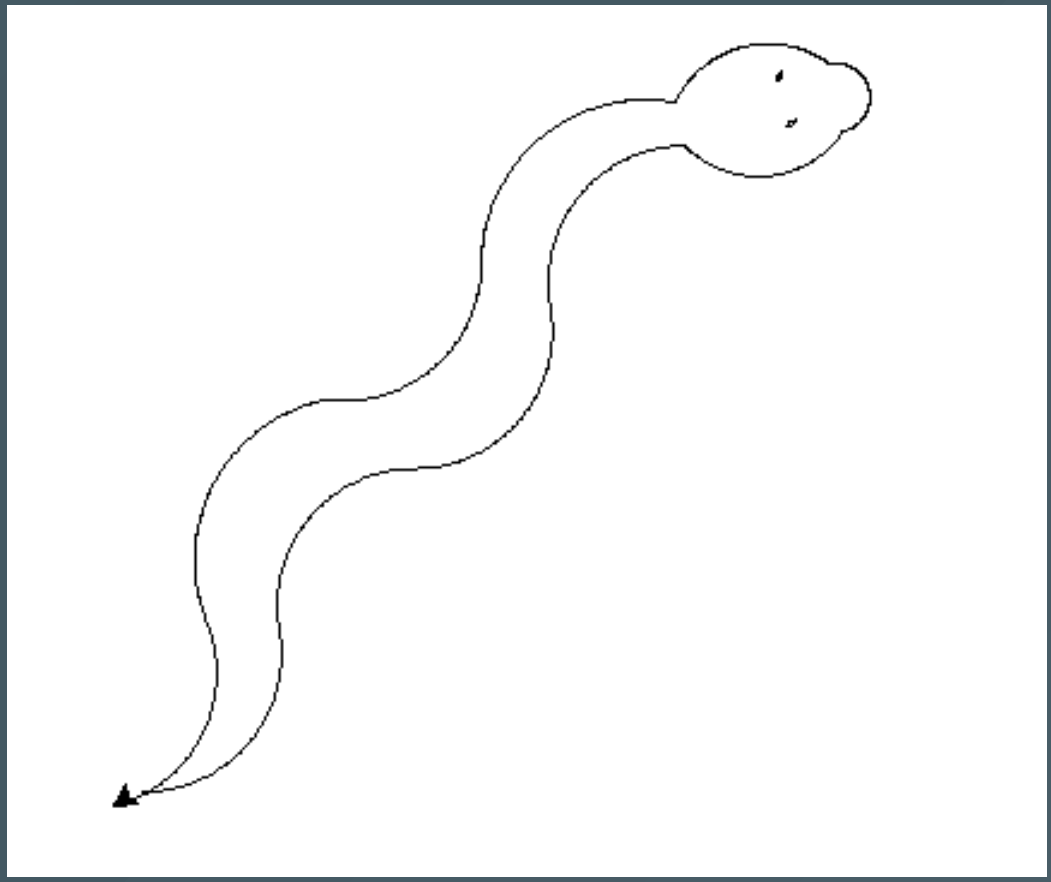

PYTHON

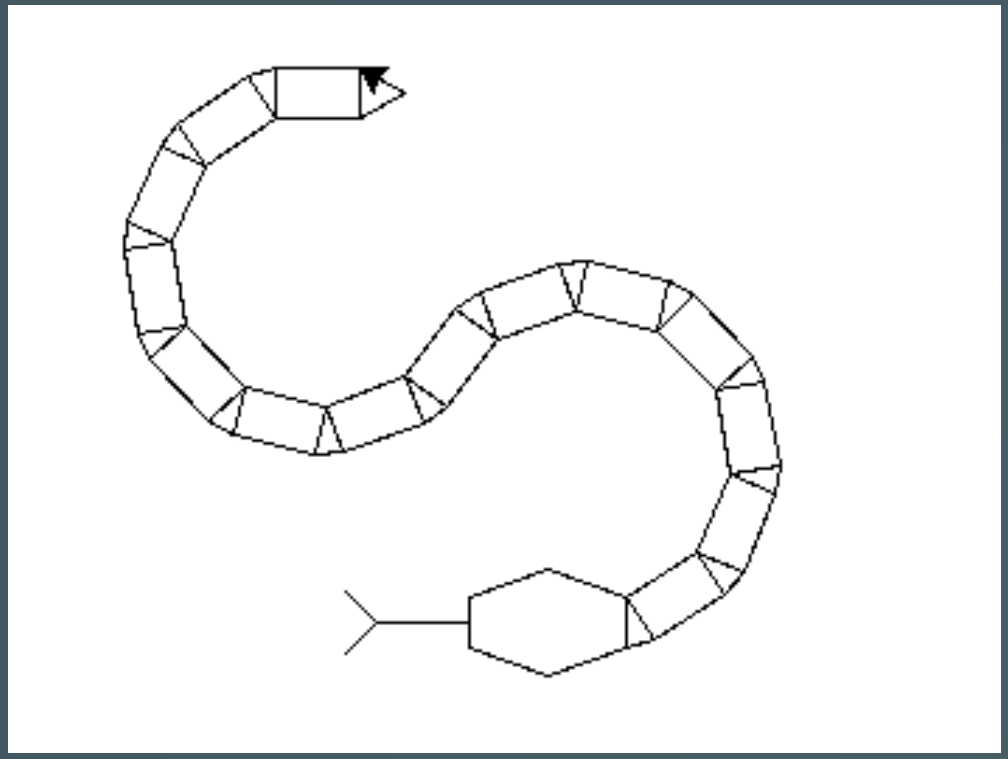


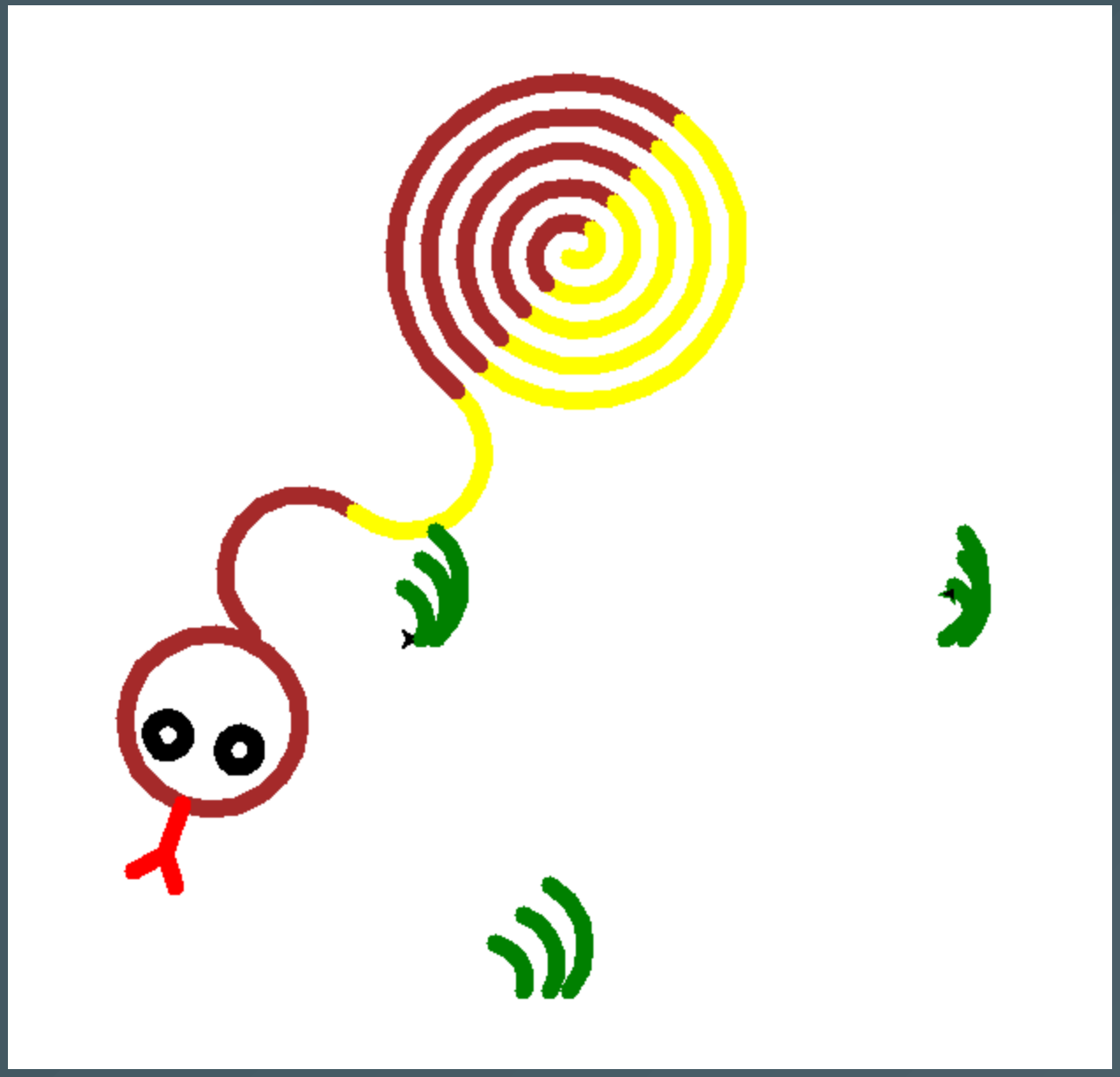


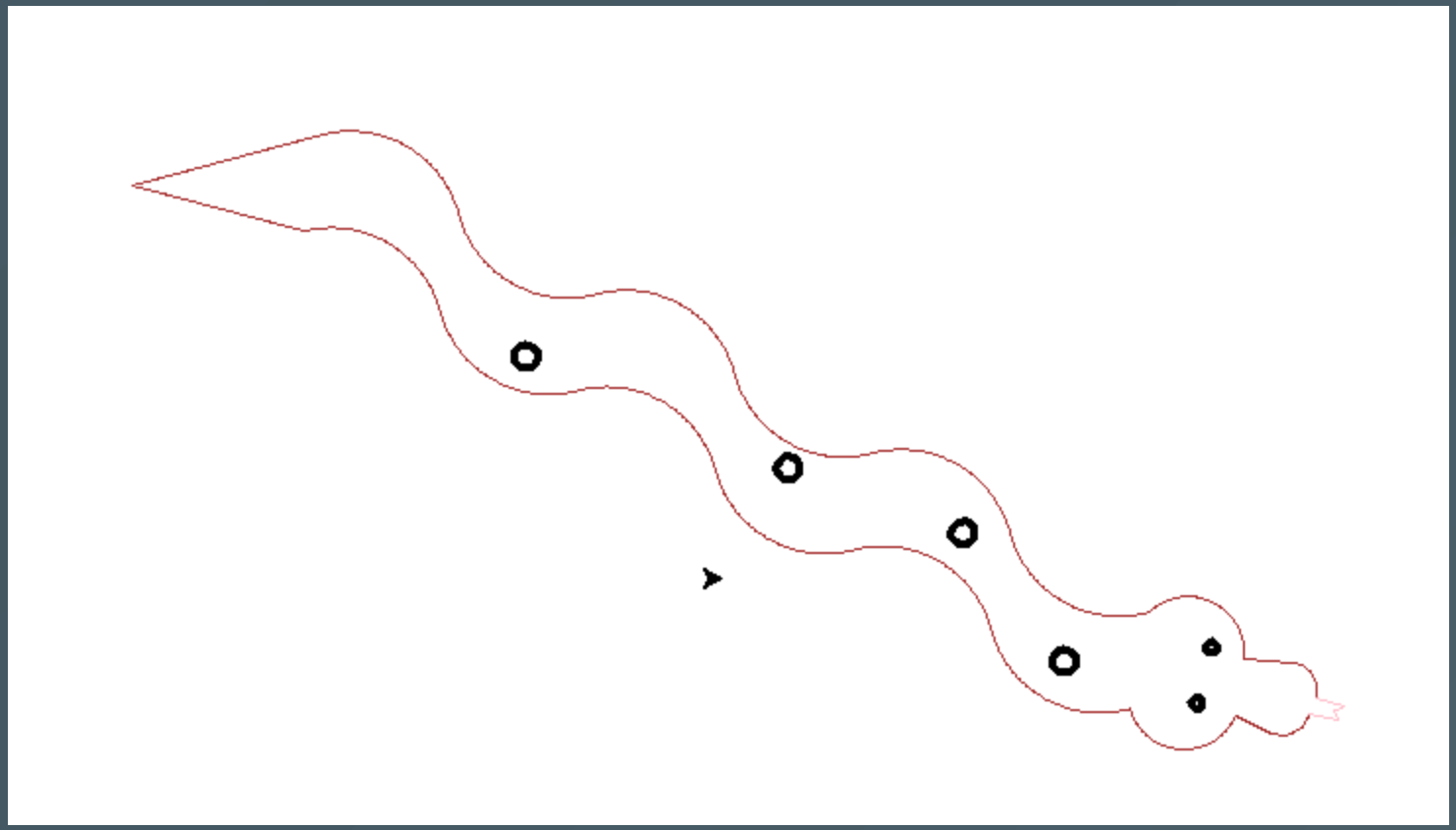


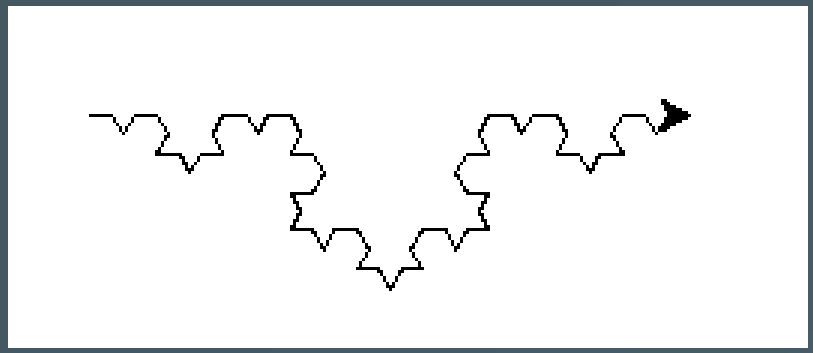


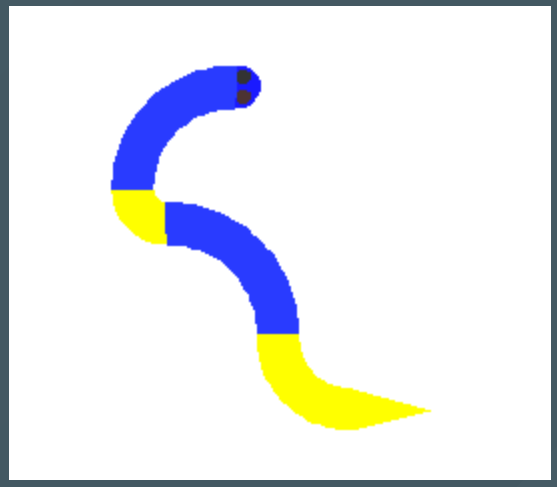


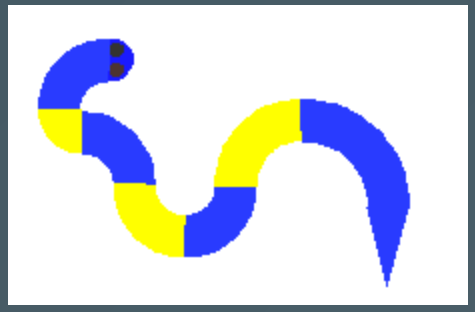


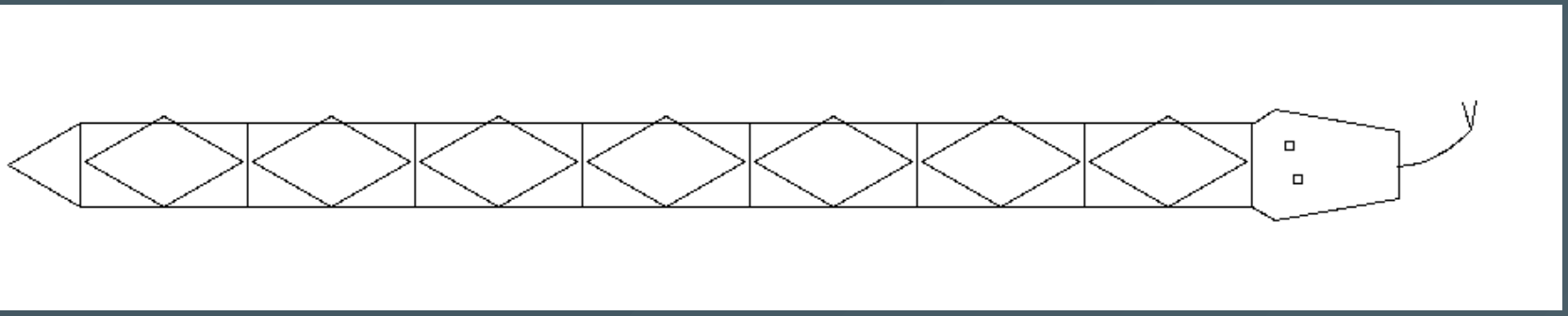


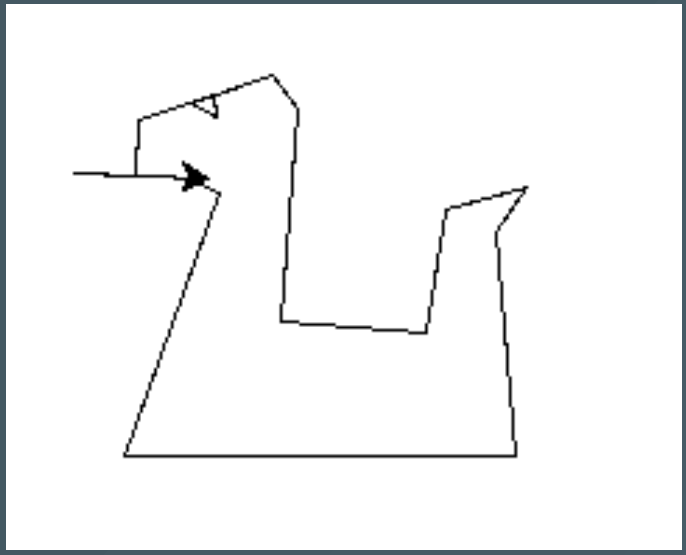


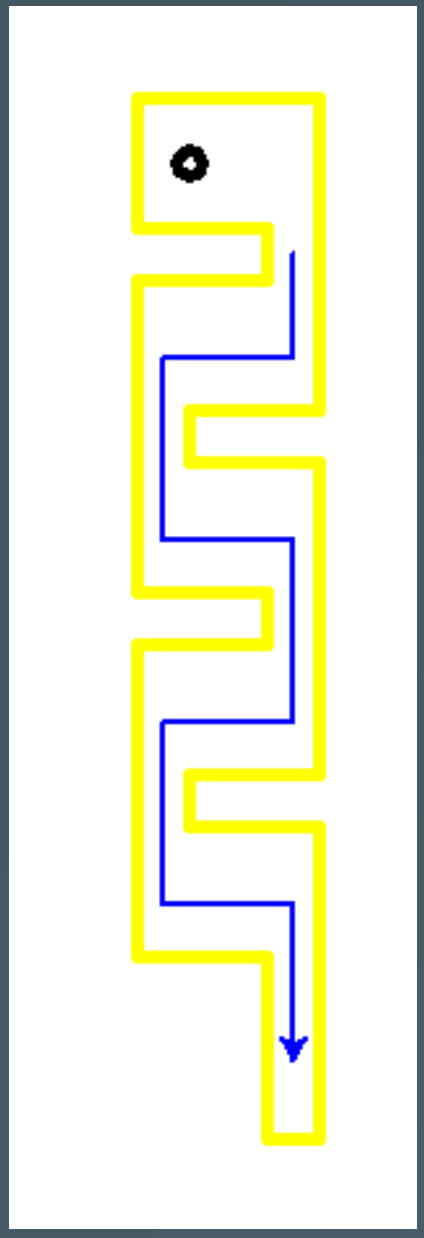












Osnova

- kontrolní otázky
- (pseudo)náhodná čísla
- simulace a nalýzy
- (debugging)
- první domácí úkol
- zadání druhé domácí úlohy