# IB111 Úvod do programovaní skrze Python


Autor: Slavomír Krupa,
Text inšpirovaný: Valdemarom Švábenským

# VNITRO


Rekurzia

"Aby ste pochopili rekurziu,
musíte pochopiť rekurziu."

Zdroj : https://en.wikipedia.org/wiki/Recursion#Recursive_humor

Zdroj: http://www.slavorum.org/matryoshka-russian-nesting-dolls/

# Rekurzia

- ★ Rekurzia je definícia funkcie (alebo dátovej štruktúry) s využitím seba samej
- ★ Zápis vždy obsahuje:
  - ○ Rekurzívne volanie
  - ○ Ukončujúcu podmienku

# Faktorial

```python
def it_fact(n):
    fact = 1
    while n > 1:
        fact *= n
        n -= 1
    return fact
```

```python
def rec_fact(n):
    if (n < 1):
        return 1
    else:
        return n * rec_fact(n - 1)
```

Nevýhoda?

# Rekurzia

```
def count_down(n):
    if n < 1:
        print("End!")
    else:
        print(n)
        count_down(n - 1)
count_down(5)
```

```
5
4
3
2
1
End!
```

# Rekurzia

```
# i = item, l = list
def all_(i, l):
    if len(l) > 0:
        return i == l[0] and all_(i, l[1:])
    else:
        return True
print(all_(3, [3,3,3]))
```

# Nepriama rekurzia

Doteraz všetko priama rekurzia

```
def even(n):
    print("even", n)
    odd(n - 1)
```

```
def odd(n):
    print("odd", n)
    if n > 1:
        even(n - 1)
```

# ?Pair programming?

# Úloha 1

- Naprogramujte funkciu využívajúcu rekurziu, ktorá sčíta prvých `n` čísel.
- `sum_of_numbers_recursive(100) => 5050`

# Úloha 2

- Naprogramujte funkciu využívajúcu rekurziu, ktorá vráti logickú hodnotu (`True`, `False`) podľa toho či zoznam obsahuje hodnotu zadanú ako parameter.
- `contains(3,[1,2]) => False`
- `contains(3,[1,3,5]) => True`
- `contains(3,[]) => False`

# Úloha 3

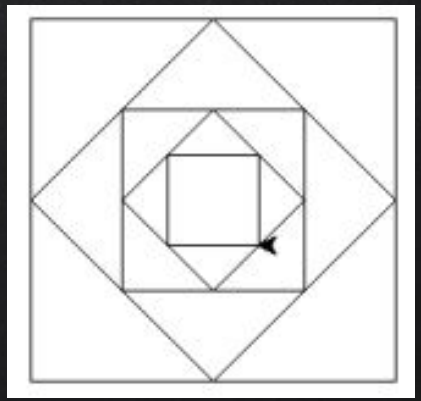
- Naprogramujte funkciu využívajúcu rekurziu, ktorá vráti “zozipsované” dva reťazce. Tzn. bude striedať znaky z prvého a druhého reťazca.

- `zip_strings("abc", "xyz") => 'axbycz'`
- `zip_strings("abc", "x") => 'axbc'`
- `zip_strings("a", "xyz") => 'axyz'`

# Úloha 4

- Naprogramujte rekurzívnu procedúru, ktorá rekurzie vykreslí zanorené štvorce. Parameter `how_many` bude špecifikovať koľko úrovní sa má vykonať

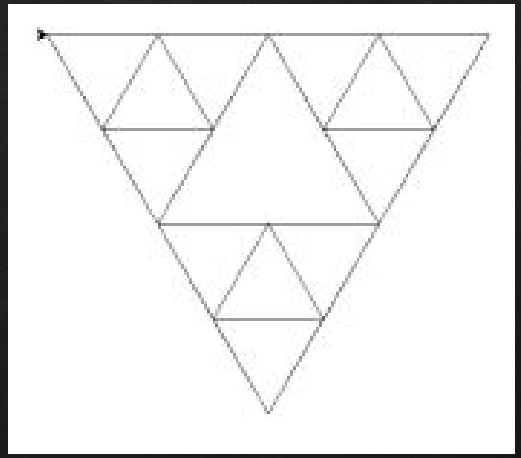
```
nested_squares(how_many=5,length=200)
```



# Úloha 5

- Naprogramujte procedúru využívajúcu rekurziu, ktorá vykreslí Sierpińského trojuholník
- Využite dekompozíciu na dve procedúry

```
sierpinski(how_many=3,length=100)
```



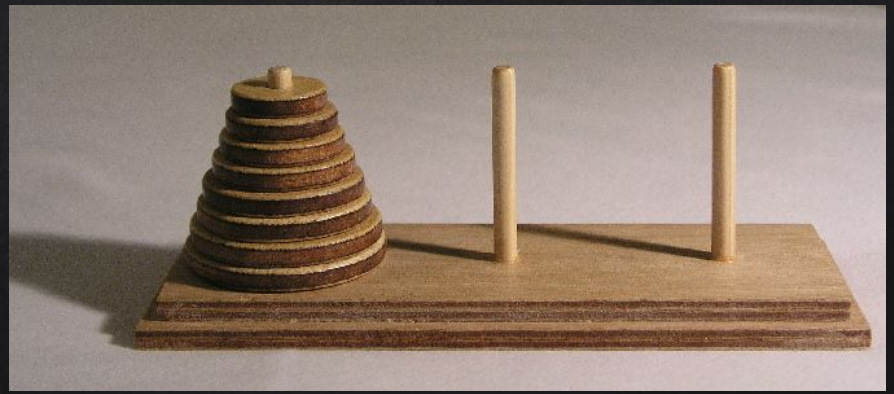
# Úloha 6

- Zadanie tejto úlohy je na viac slidov
- Pred riešením si prečítajte všetky slidy pre úlohu 6

- Máme 3 rovnaké kolíky a `n` diskov, všetky rôznych veľkostí.
- Na začiatku sú disky naskladané v poradí od najväčšieho na 1. kolíku (a tvoria akýsi kužeľ)
- Úlohou je presunúť celý “kužeľ” diskov na 3. kolík

# Úloha 6

1. Naraz môžeme hýbať len jedným diskom
2. V každom ťahu môžeme vziať nejaký disk z vrchu hromady a presunúť ho na vrch inej hromady
3. Žiadny disk nesmie byť položený na menší disk



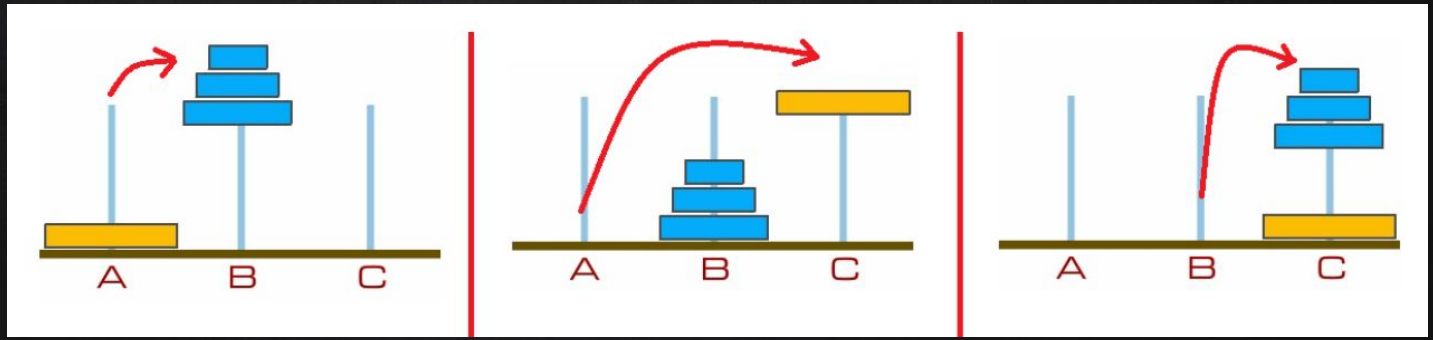
Zdroj: https://en.wikipedia.org/wiki/File:Tower_of_Hanoi.jpeg

# Úloha 6

Označme kolíky písmenami `A`, `B`, `C`

Pre presun `n` diskov z kolíku `A` na kolík `C`:

1. Presuň `n-1` diskov z `A` na `B`.
   a. Disk `n` ostane sám na `A`.
2. Presuň `1` (najväčší) disk z `A` na `C`.
3. Presuň `n-1` diskov z `B` na `C` (na najväčší disk).

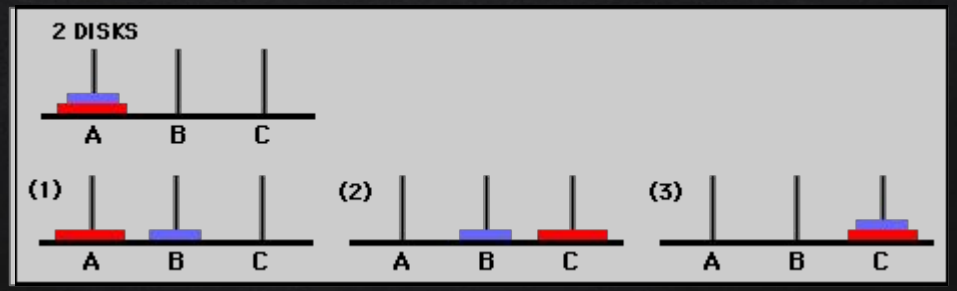



Zdroj: http://www.fi.muni.cz/~xpelanek/IB111/slidy/rekurze.pdf

# Úloha 6

Napíšte funkciu `hanoi(n,from_pole,to_pole,helper)`, ktorá vypíše kroky presunu `n` diskov z kolíku `from_pole(A)` na kolík `to_pole(C)` za pomoci kolíka `helper(B)`.

```
>>>hanoi (2,"A","C","B")
A -> B
A -> C
B -> C
```




Zdroj: http://mathforum.org/dr.math/faq/faq.tower.hanoi.html

# Úloha 1*

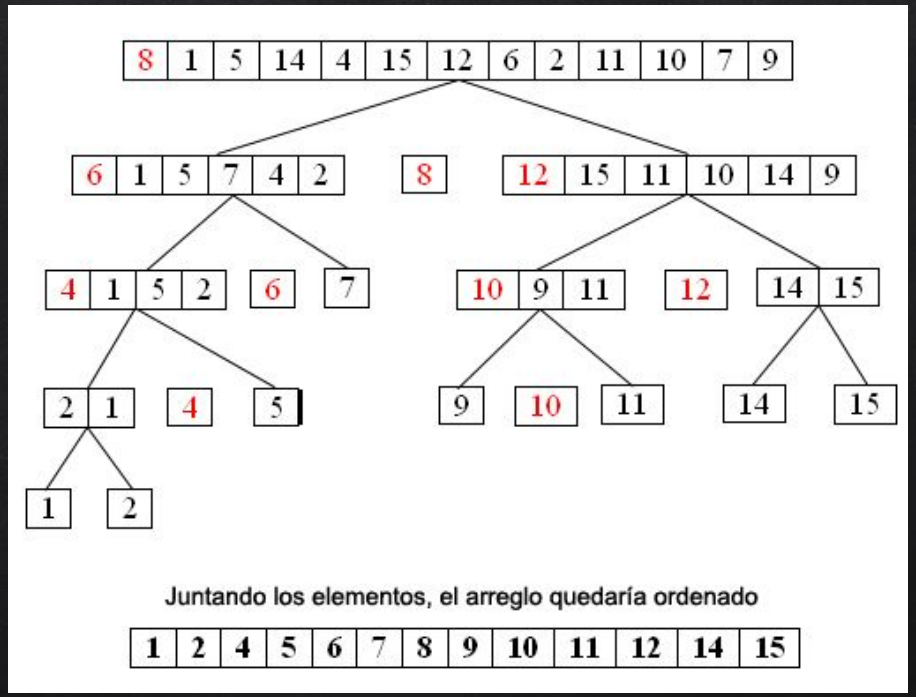
- Naprogramujte funkciu využívajúcu rekurziu, ktorá vráti zoznam hodnôt zoznamu menších ako zadaný parameter
- `list.insert(0,i)` - vloží premennú i na prvú pozíciu
- `smaller_than(3,[1,2]) = [1,2]`
- `smaller_than(6,[1,5,3]) = [1,5,3]`
- `smaller_than(100,[]) = []`

# Quick sort**

- Vstup: množina čísel S

```
def qsort(S):
    if len(S) <= 1:
        return S
    randomly pick pivot x from S
    L = {y in S | y < x}
    R = {y in S | y > x}
    return qsort(L) + {x} + qsort(R)
```

# Quick sort**




Zdroj: https://devtalk.nvidia.com/default/topic/496167/need-help-understanding-this-function-__device_

# Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash