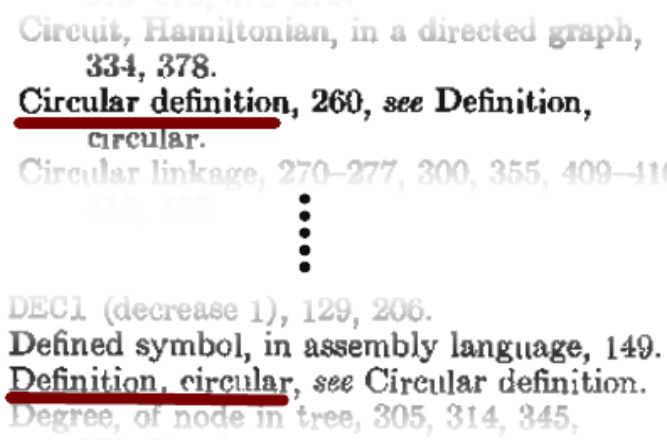
Rekurzia

Rekurzia



Obr. 1: Z knižky *The Art of Programming* od D. Knuth

```
def rekurzia():
    rekurzia()
```

Poznámka 1: V Pythone je rekurzia obmedzená a väčšinou sa preferuje `for`-cyklus (naučíme sa na ďalšej hodine).

Poznámka 2: `Ctrl + C` je váš kamarát, ak by ste sa niekedy dostali do problémov (alebo tlačítko “Stop” v jupyteru, príp. Kernel -> Restart).

Rekurzia v praxi

Napíšte funkciu, ktorá odpočíta od daného čísla do 0.

```
def countdown(n):
    if n == 0:
        print(0)
    else:
        print(n)
        countdown(n - 1)
```

Každá užitočná rekurzívna funkcia niekde skončí

- Rekurzívne funkcie budú mať `if` príkaz.
- Jedna vetva `if` obsahuje tzv. *base case* -> žiadna rekurzia / koniec rekurzie
- Druhá vetva obsahuje rekurziu so zmeneným argumentom.

```
def countdown(n):
    if n == 0:          # base case
        print(0)
    else:               # rekurzivna vetva
        print(n)
        countdown(n - 1) # zmeneny argument
```

Zjednodušenie

- `print(n)` nastane v oboch vetvách

```
def countdown(n):
    print(n)
    if n > 0:
        countdown(n - 1)
```

Cvičenie 1

Pomocou rekurzie a *if* podmienok, napíšte funkciu, ktorá:

- vypíše všetky párne (sudé) čísla od n do 0.

Ďalšie príklady

Čo počíta nasledujúca funkcia?

```
def what_do_i_do_huh(a, b):
    if b == 0:
        return a
    elif b > 0:
        return what_do_i_do_huh(a + 1, b - 1)
    else:
        return what_do_i_do_huh(a - 1, b + 1)
```

Ďalšie príklady

Čo vypíše nasledujúca funkcia?

```
def what_do_i_do_huh(n):
    if n == 1:
        print(n - 1)
    return what_do_i_do_huh(n - 1)

what_do_i_do_huh(10)
```

Ďalšie príklady

Čo vypíše nasledujúca funkcia?

```
def what_do_i_do_huh(n):
    if n == 3:
        print(n)
    elif n > 0:
        return what_do_i_do_huh(n - 1)
    else:
        return

what_do_i_do_huh(32)
what_do_i_do_huh(3)
what_do_i_do_huh(1)
what_do_i_do_huh(4.3)
```

Ďalšie príklady

Čo vypíše nasledujúca funkcia?

```
def test(n):
    print(n)
    if n > 1:
        test(n - 1)
    print(-n)
```

Cvičenie 2

Prečitajte si nasledujúci kód a snažte sa zistiť ako bude výsledný obrázok vyzerať:

```
import turtle

def draw(t, length, n):
    if n == 0:
        return
    angle = 50
    t.forward(length * n)
    t.left(angle)
    draw(t, length, n-1)
    t.right(2 * angle)
    draw(t, length, n-1)
    t.left(angle)
    t.backward(length * n)


stromozrava_korytnacka = turtle.Turtle()
stromozrava_korytnacka.speed(1)

n = 5
l = 50 / n

draw(stromozrava_korytnacka, l, n)

# input()     # pocka na to az stacite Enter (nezavrie okno)
```

Potom si túto funkciu vyskúšajte pre rôzne `n` a vysvetlite ju.

Cvičenie 3

Napíšte rekurzívnu funkciu, ktorá počíta faktoriál z čísla `n`.

Malá matematická nápoveda:

```
0! = 1
n! = n(n-1)!
```

napr. 5! = 5 * 4 * 3 * 2 * 1 * 1 = 120

Cvičenie 4

Napíšte rekurzívnu funkciu, ktorá počíta `n`-té číslo Fibonacciho postupnosti.

Malá matematická nápoveda:

```
fibonacci(0) = 0
fibonacci(1) = 1
fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)
```

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}
0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597

Obr. 2: Začiatok postupnosti do n=17

Demo - Hanojské veže

- inšpirované videom
- anglicky Tower of Hanoi
- objekty a metódy na vykresľovanie
- jupyter notebook demo v študijných materiáloch

Teraz už naprogramujete všetko... teoreticky

Turingovká úplnosť

Tiež, niektoré veci dokázateľne nie je možné vyriešiť programom -> viz. halting problem

Cvičenie 5

- Príklad 5.6 z Think Python, časť 1 a 2. (koch)
- Príklady 6.4 (is_power) a 6.5 (gcd) z Think Python.

Cvičenie 6

Príklad 6.2 z Think Python. Je možné, aby pre nejaké hodnoty `m` a `n` funkcia nikdy (akože fakt nikdy - bez ohľadu na život vašeho počítača alebo vesmíru) nedobehla?

Viac o Ackermannovej funkcií v tomto videu.

Cvičenie 7

- Napíšte rekurzívnu funkciu, ktorá vyhodnotí sumu:

$$S(n) = \sum_{i=0}^{n} \sin(i) \tag{1}$$

- Túto funkciu zobecnite, aby brala dodatočný argument, a to funkciu, z ktorej sa má počítať suma:

$$S(n, f) = \sum_{i=0}^{n} f(i) \tag{2}$$