MUNI
SCI

# Pokročilé funkcie, rekurzia

Kryštof Mrózek (445429@mail.muni.cz)
Kristína Tomanková (kristinatomankova@mail.muni.cz)
Radoslav Brunovský (rbrunovsky@mail.muni.cz)

# Rekurzia

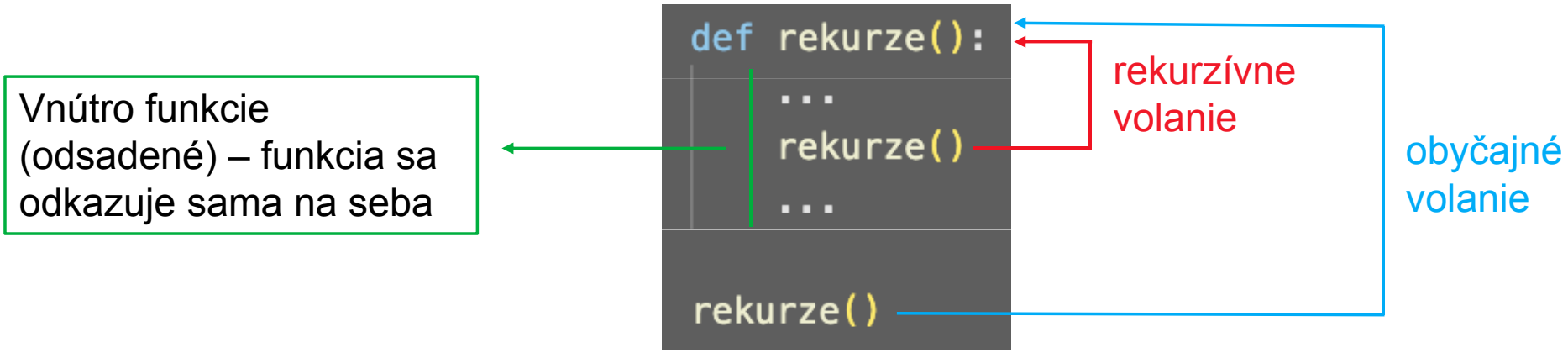
– Rekurzia je proces, kedy sa objekt definuje pomocou samého seba.



MUNI SCI

# Rekurzívna funkcia

– Funkcia, ktorá volá sama seba.




**Spôsobí takýto zápis funkcie nejaký problém?**



MUNI SCI

# Základný prípad (base case)

- Každá rekurzívna funkcia musí mať tzv. “base case“, ktorý po určitej dobe **ukončí jej volanie**.
- Pokiaľ funkcia ničím neskončí, tak python ju skúsi zavolať max 1000x, pokým nevyhodí RecursionError:

```
def rekurze():
    rekurze()

rekurze()
```

1.2s

```
RecursionError: maximum recursion depth exceeded
```



MUNI SCI

# Základný prípad (base case)

- Každá rekurzívna funkcia musí mať tzv. “base case“, ktorý po určitej dobe **ukončí jej volanie**.
- Príklad: Spočítajte všetky čísla od 1 po x

```
x = 4
vysledok = 0

for i in range(x+1):
    vysledok += i

print(vysledok)
```

✓ 0.0s

```
10
```



MUNI SCI

# Základný prípad (base case)

- Každá rekurzívna funkcia musí mať tzv. “base case“, ktorý po určitej dobe **ukončí jej volanie**.
- Príklad: Spočítajte všetky čísla od 1 po x

Base case – zadaný pomocou if podmienky

```
def suma(x):
    # Tato funkcia spocita vsetky cisla od 1 po x
    if x == 1:
        return 1
    else:
        return x + suma(x-1)
```

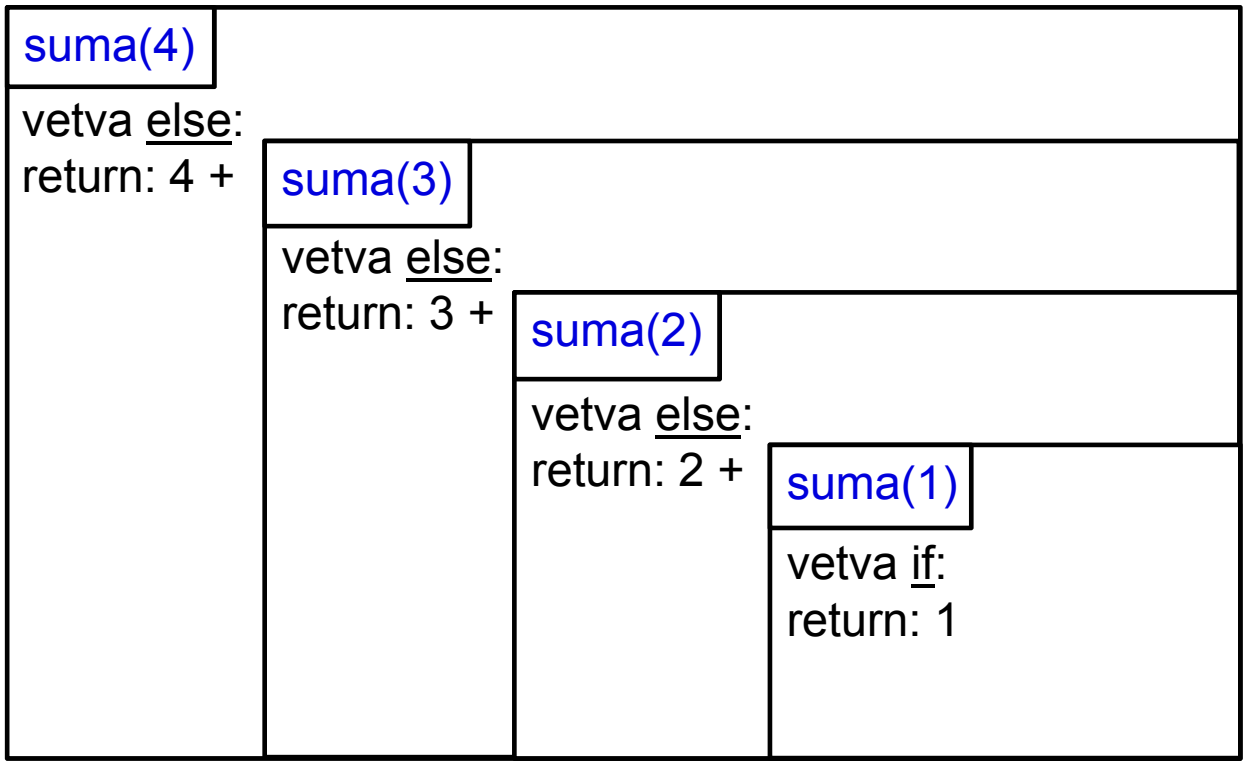


MUNI SCI

# Základný prípad (base case)

*Čo sa reálne deje, keď zavolám suma(4):*

```
def suma(x):
    # Tato funkcia spocita vsetky cisla od 1 po x
    if x == 1:
        return 1
    else:
        return x + suma(x-1)
```

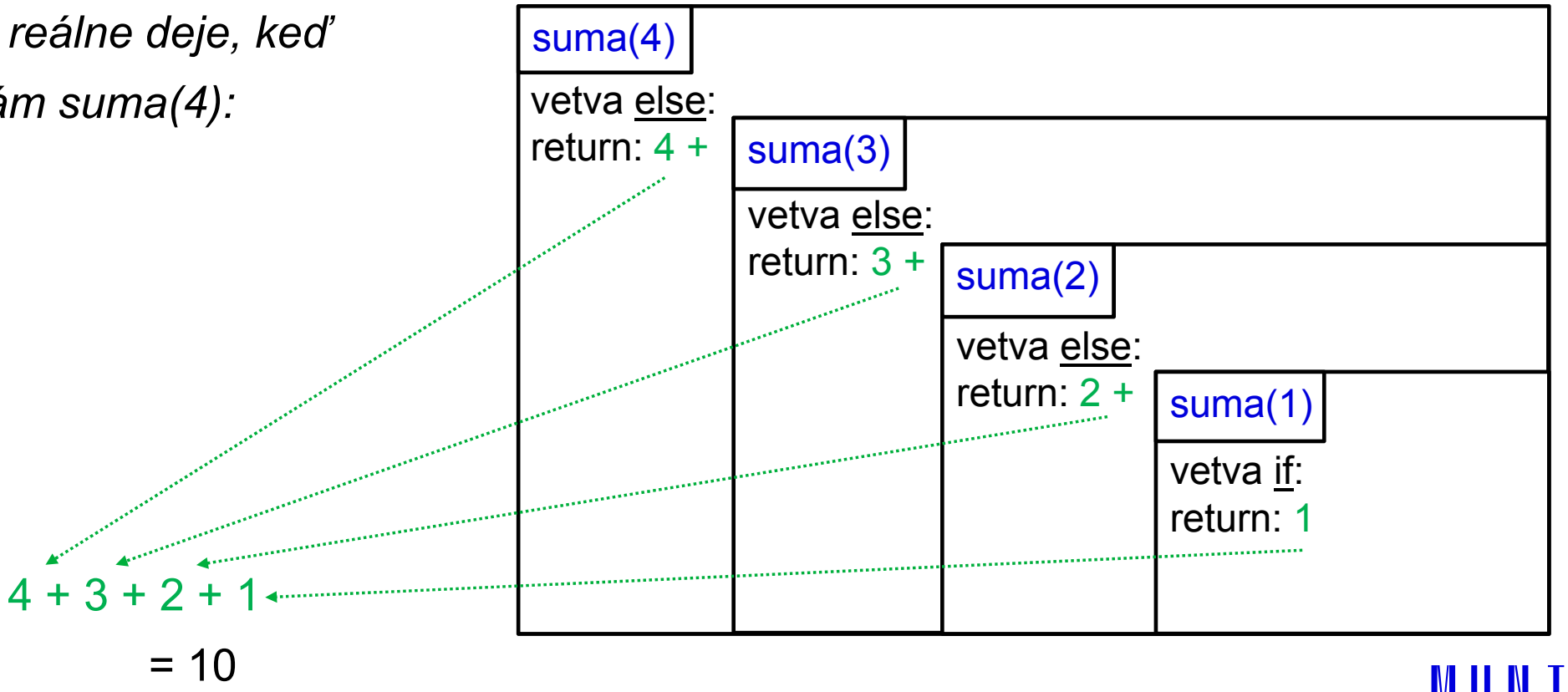





MUNI SCI

# Základný prípad (base case)

*Čo sa reálne deje, keď zavolám suma(4):*






MUNI SCI

# Cvičenie

- Napíšte rekurzívnu funkciu, ktorá spočíta faktoriál daného celého čísla. Uistite sa, že vaša funkcia podchytí nasledujúce fenomény:
  - Kladné číslo x: $x! = x \cdot (x-1) \cdot (x-2) \ldots 2 \cdot 1$
  - Nula: $0! = 1$
  - Záporné číslo: Faktoriál neexistuje

# Cvičenie

– Napíšte rekurzívnu funkciu, ktorá spočíta faktoriál daného celého čísla.

```
def factorial(x):
    if x == 0:
        return 1
    elif x < 0:
        return None
    else:
        return x * factorial(x-1)

print(factorial(-4))
```

✓ 0.0s

```
None
```

Viacero základných prípadov



MUNI SCI

# Cvičenie

– Napíšte rekurzívnu funkciu, ktorá spočíta faktoriál daného celého čísla.

```
def factorial(x):
    if x == 0:
        return 1
    elif x < 0:
        raise ValueError('Factorial of a negative number is not defined!')
    else:
        return x * factorial(x-1)

print(factorial(-4))
```

⊗ 0.0s

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[23], line 9
      6     else:
      7         return x * factorial(x-1)
----> 9 print(factorial(-4))

Cell In[23], line 5, in factorial(x)
      3     return 1
      4 elif x < 0:
----> 5     raise ValueError('Factorial of a negative number is not defined!')
      6 else:
      7     return x * factorial(x-1)

ValueError: Factorial of a negative number is not defined!
```



MUNI SCI

# Rekurzívne funkcie vs iterácie

- Rekurziu môžeme chápať aj ako “elegantnú“ iteráciu (všetky rekurzívne problémy sa dajú riešiť aj iterovaním).
- Príklad: Faktoriál

```
def factorial(x):
    if x == 0:
        return 1
    elif x < 0:
        return None
    else:
        return x * factorial(x-1)
```

**Rekurzia**

```
def fact_iterace(x):
    if x < 0:
        return None
    else:
        vysledok = 1
        for i in range(1, x+1):
            vysledok *= i
        return vysledok
```

**Iterácia**



MUNI SCI

# Rekurzívne funkcie vs iterácie

- Rekurziu môžeme chápať aj ako “elegantnú“ iteráciu (všetky rekurzívne problémy sa dajú riešiť aj iterovaním).
- Príklad: Faktoriál

```python
def factorial(x):
    if x == 0:
        return 1
    elif x < 0:
        return None
    else:
        return x * factorial(x-1)

print(factorial(50))
```

```python
def fact_iterace(x):
    if x < 0:
        return None
    else:
        vysledok = 1
        for i in range(1, x+1):
            vysledok *= i
        return vysledok

print(fact_iterace(50))
```

**Výpočetný čas:** $2.8 \cdot 10^{-4}$ s vs $5.01 \cdot 10^{-6}$ s

MUNI SCI

# Rekurzívne funkcie vs iterácie

- Rekurziu môžeme chápať aj ako “elegantnú“ iteráciu (všetky rekurzívne problémy sa dajú riešiť aj iterovaním).
- Príklad: Faktoriál – v prípade volania funkcie na číslo 50 dostaneme výsledok v prípade využitia **iterácie cca 56x rýchlejšie**.

| Výhody rekurzie | Nevýhody rekurzie |
|---|---|
| Elegantný a prehľadný kód | Rekurzívne volanie funkcie zaberá veľa pamäte a času |
| Komplexný problém je rozdelený do jednoduchých pod-problémov | Náročné na debug |

* *Rekurziu je výhodné použiť v prípadoch, kde je vnáranie logické a prirodzené – napr. pri navigovaní v súboroch na počítači.*

# Záverečné cvičenie

– Predstavte si matriošku, ktorá má veľkosť x (predstavuje výšku matriošky). Každá nasledujúca matrioška je o 20% menšia ako tá predchádzajúca. Minimálna veľkosť matriošky je 3 cm. Napíšte funkciu, ktorá spočíta koľko matriošiek celkom obsahuje matrioška o veľkosti x.



MUNI SCI