

 a cykly  
E 3011

Jan Böhm

RECETOX

March 1, 2023

# Co nás dnes čeká



2 For cyklus v Pythonu

3 True & False

4 While cyklus v Pythonu

## Iterace

Iterace je fancy slovo pro opakování. Častým úkolem je provést stejnou sérii úkonů pro každou hodnotu z nějakého seznamu nebo dokud nedojde k naplnění nějaké podmínky.

## Iterace

Iterace je fancy slovo pro opakování. Častým úkolem je provést stejnou sérii úkonů pro každou hodnotu z nějakého seznamu nebo dokud nedojde k naplnění nějaké podmínky.

## Příklady

- Spočítat *pro každého* pacienta popisné statistiky.
- *Pro každou* hodnotu v matici rozhodnout, zda je kladná.
- *Pro každý* index nějaké posloupnosti určit částečný součet této posloupnosti.
- Počítat částečné součty dané posloupnosti *dokud* nestanovím celkový součet s danou tolerancí.
- Dělení: odečítej od dělence dělitele, *dokud* nedosáhneš nuly.

# For cyklus

---

## Algorithm 1: Tajemný algoritmus I

---

**Input:**  $A = \{a_i\}_{i=1}^n$  posloupnost délky  $n$

$s \leftarrow 0$  ;

**for**  $a \in A$  **do**

  |  $s \leftarrow s + a$

**end**

print( $s$ );

---

## Otázky

Nechť  $A = \{3; 8; 0; 7; 2; -10; 6\}$ .

- Kolik je  $n$ ?
- Jak se mění postupně hodnota proměnné  $s$ ?
- Co nakonec vypíše tento algoritmus?
- Co dělá tento algoritmus?
- Co by algoritmus vypsal pro  $A = \emptyset$ ?

# For cyklus v Pythonu

---

## Algorithm 2: Tajemný algoritmus I

---

**Input:**  $A = \{a_i\}_{i=1}^n$  posloupnost délky  $n$

$s \leftarrow 0$  ;

**for**  $a \in A$  **do**

$s \leftarrow s + a$

**end**

print( $s$ );

---

```
1 A = [3, 8, 0, 7, 2, -10, 6]
2 s = 0
3 for a in A:
4     s = s + a
5 print(s)
```

### Algorithm 3: Tajemný algoritmus II

**Input:**  $A = \{a_i\}_{i=1}^n$  posloupnost délky  $n$

$m \leftarrow \infty$ ;

$M \leftarrow -\infty$ ;

**for**  $a \in A$  **do**

**if**  $a > M$  **then**

$m \leftarrow a$

**end**

**if**  $a < m$  **then**

$M \leftarrow a$

**end**

**end**

print( $m, M$ );

### Otázky

Nechť  $A = \{3; 8; 0; 7; 2; -10; 6\}$ .

- Vypište posloupnost, jak se mění hodnoty  $m$  a  $M$ .
- Co nakonec vypíše tento algoritmus? Co je  $m$  a  $M$ ?

# Co nás dnes čeká



1

2 For cyklus v Pythonu

3 True & False

4 While cyklus v Pythonu



## Nejčastější konstrukce

Cykly budeme psát pořád. Projdeme si několik konstrukcí, které se budou hodit

- 1 Pro hodnoty z nějaké posloupnosti  $[a_1, a_2, \dots, a_n]$
- 2 Pro každý znak z nějakého slova "python"
- 3 Pro index  $ind = \underbrace{0, 1, \dots, n-1}_{\text{celkem } n \times} = \text{range}(n)$ .

Situaci 1) už jsme viděli. Ukážeme si zbylé 2.

## Tento kód

```
1 UCO = "408849" #number as string
2 s = 0
3 for cifra in UCO:
4     s += int(cifra)
5 print(s)
```

## a tento kód

```
1 UCO = "408849" #number as a string
2 s = 0
3 n = len(UCO) #len - abbr. length (number of characters of
4     string)
5 for ind in range(n):
6     s += int(UCO[ind])
7 print(s)
```

fungují stejně. Ale co dělají?

# Co nás dnes čeká



2 For cyklus v Pythonu

3 True & False

4 While cyklus v Pythonu

# True & False

## True & False

V Pythonu (a obecně během programování) se budeme setkávat se dvěma pravdivostními (Boolean) hodnotami – pravdou True a nepravdou False.

# True & False

## True & False

V Pythonu (a obecně během programování) se budeme setkávat se dvěma pravdivostními (Boolean) hodnotami – pravdou True a nepravdou False.

## Logické spojky

Prosté True a False nám nestačí. Občas potřebujeme vyhodnotit víc logických výrazů za sebou, proto potřebujeme logické spojky:

- `and` pro konjunkci
- `or` pro alternativu
- `not` (případně `!`) pro negaci

Python obsahuje také `&` pro bitwise `and` a `|` pro bitwise `or`, které se od toho "klasického" liší.

# True & False

Zamyslete se nad následujícími výrazy a zkuste je v konzoli vyhodnotit:

```
1 True and True
2 True and not True
3 False or True
4 True and (False or True)
5 not(not True or False)
```

## Vyhodnocování logických výrazů

Často budeme potřebovat rozhodnout o nějčém výrazu, zda platí či ne – získat hodnotu `True` nebo `False`. Nejčastější konstrukce jsou:

- test rovnosti `a == b`
- test, zda se dvě hodnoty liší `a != b`
- test ostré nerovnosti `a < b`
- test neostré nerovnosti `a <= b`

## Vyhodnocování logických výrazů

Často budeme potřebovat rozhodnout o nějakém výrazu, zda platí či ne – získat hodnotu `True` nebo `False`. Nejčastější konstrukce jsou:

- test rovnosti `a == b`
- test, zda se dvě hodnoty liší `a != b`
- test ostré nerovnosti `a < b`
- test neostré nerovnosti `a <= b`

Zkuste vyhodnotit (nejprve v hlavě, poté pomocí konzole) logické výrazy:

```
1 0 < 1 and 1 <= 2
2 5 != 5 or 3 < 2
3 1 < 2 and 2 == 3
```



## Tipy a triky

- Lze řetězit nerovnosti  $1 < x$  and  $x < 2$  je totéž jako  $1 < x < 2$
- Lze použít `in`, například `3 in [1,2,3,4,5]` nebo `"j" not in "Jan"`

## Tipy a triky

- Lze řetězit nerovnosti  $1 < x$  and  $x < 2$  je totéž jako  $1 < x < 2$
- Lze použít `in`, například `3 in [1,2,3,4,5]` nebo `"j" not in "Jan"`

## Časté chyby

- Špatně napsaný `True` nebo `False`, hlavně velké první písmeno.
- Záměna přiřazení `a = b` za test rovnosti `a == b`
- Když si nejsem jistý prioritou, použiji závorky.

# Co nás dnes čeká



2 For cyklus v Pythonu

3 True & False

4 While cyklus v Pythonu

# While cyklus

---

## Algorithm 4: Tajemný algoritmus III

---

**Input:**  $n \in \mathbb{N}$

$k \leftarrow 1$ ;

**while**  $k < n$  **do**

$k \leftarrow 2 * k$

**end**

print( $k$ );

---

## Otázky

Nechť  $n = 100$ .

- Jak se mění postupně hodnota proměnné  $k$ ?
- Co nakonec vypíše tento algoritmus?
- Kolik iterací kódu ve while bloku proběhne?
- Jak proběhne algoritmus pro  $n = 0$ ?

# While cyklus v Pythonu

---

## Algorithm 5: Tajemný algoritmus III

---

**Input:**  $n \in \mathbb{N}$

$k \leftarrow 1$ ;

**while**  $k < n$  **do**

$k \leftarrow 2 * k$

**end**

`print(k)`;

---

```
1 n = 100
2
3 k = 1
4 while k < n:
5     k = 2 * k
6 print(k)
```

# While cyklus v Pythonu

---

## Algorithm 6: Tajemný algoritmus IV

---

**Input:**  $D \in \mathbb{N}$ ;  
 $d \in \mathbb{N}$ ;

$p \leftarrow 0$ ;

**while**  $d < D$  **do**

$D \leftarrow D - d$ ;  
     $p \leftarrow p + 1$ ;

**end**

print( $p, D$ );

---

## Úkoly

Nechť  $D = 100$  a  $d = 12$ .

- Jak se mění postupně hodnoty  $D, d, p$ ?
- Co nakonec vypíše tento algoritmus?
- Jaký proces kód popisuje?