

# Cykly

Poznámky k téme

Počítače sú silné v tom, že vedia tú istú činnosť zopakovať tisíckrát bez chyby. Opakovanie sa v programe zapisuje **cyklom**. Python má dva druhy cyklov: `for` – keď vieme, koľkokrát (alebo pre ktoré hodnoty) sa má niečo zopakovať, a `while` – keď sa má opakovať, kým platí podmienka.

## Čo sa naučíš

- opakovať príkazy cyklom `for` s funkciou `range`,
- použiť cyklus `while` a vyhnúť sa nekonečnému cyklu,
- počítat pomocou počítadla a akumulátora (súčet, súčin, priemer),
- predčasne ukončiť cyklus (`break`) alebo preskočiť krok (`continue`),
- písať vnorené cykly a trasovať, čo cyklus robí.

## 1 Cyklus for a funkcia range

```
for i in range(5):  
    print("Opakovanie číslo", i)  
print("Hotovo")
```

Výstup:

```
Opakovanie číslo 0  
Opakovanie číslo 1  
Opakovanie číslo 2  
Opakovanie číslo 3  
Opakovanie číslo 4  
Hotovo
```

Ako to funguje:

- `range(5)` vytvorí čísla 0, 1, 2, 3, 4 – **päť** čísel, začína sa nulou a **5 už nepatrí** do výsledku.
- Premenná `i` (*riadiaca premenná cyklu*) postupne nadobudne každú z týchto hodnôt.
- Pre každú hodnotu sa vykoná **telo cyklu** – odsadený blok pod riadkom s dvojbodkou.
- Neodsadený riadok `print("Hotovo")` sa vykoná až po skončení cyklu, raz.

### 1.1 Tri podoby range

| Zápis                            | Čísla               | Význam                       |
|----------------------------------|---------------------|------------------------------|
| <code>range(n)</code>            | 0, 1, ..., n-1      | n čísel od nuly              |
| <code>range(od, do)</code>       | od, od+1, ..., do-1 | <code>do</code> už nepatrí   |
| <code>range(od, do, krok)</code> | od, od+krok, ...    | kým sme pred <code>do</code> |

```
>>> list(range(5))
[0, 1, 2, 3, 4]
>>> list(range(2, 8))
[2, 3, 4, 5, 6, 7]
>>> list(range(1, 11, 3))
[1, 4, 7, 10]
>>> list(range(10, 0, -2))
[10, 8, 6, 4, 2]
>>> list(range(5, 1))
[]
```

(`list(...)` tu iba ukazuje, ktoré čísla `range` vytvorí.) So záporným krokom sa počíta smerom dole; ak je od už „za“ hranicou, `range` je prázdny a telo cyklu sa nevykoná ani raz.

**Pozor:** Chceš čísla od 1 do n vrátane? Hornú hranicu zväčši o 1: `range(1, n + 1)`. Zabudnutá jednotka je najčastejšia chyba pri cykloch – hovorí sa jej chyba „o jednu“ (*off-by-one*).

## 1.2 Riadiaca premenná v tele cyklu

```
for i in range(1, 6):
    print(i, "na druhú je", i ** 2)
```

Výstup:

```
1 na druhú je 1
2 na druhú je 4
3 na druhú je 9
4 na druhú je 16
5 na druhú je 25
```

Ak hodnotu riadiacej premennej nepotrebuješ, zvykne sa pomenovať `_`:

```
for _ in range(3):
    print("Hip hip hurá!")
```

Výstup:

```
Hip hip hurá!
Hip hip hurá!
Hip hip hurá!
```

## 1.3 Výpis do riadku

Vďaka `end=" "` vypíšeš hodnoty z cyklu vedľa seba. Prázdny `print()` po cykle ukončí riadok:

```
for i in range(10, 0, -1):
    print(i, end=" ")
print()
print("Štart!")
```

Výstup:

```
10 9 8 7 6 5 4 3 2 1
Štart!
```

## 1.4 for cez text a zoznam

Cyklus `for` vie prechádzať aj znaky reťazca alebo prvky zoznamu – nielen čísla z `range`:

```
for znak in "Python":  
    print(znak, end="-")  
print()  
for ovocie in ["jablko", "hruška", "slivka"]:  
    print("Ovocie:", ovocie)
```

Výstup:

```
P-y-t-h-o-n-  
Ovocie: jablko  
Ovocie: hruška  
Ovocie: slivka
```

Viac v témach Reťazce a Zoznamy.

## 2 Cyklus while

Cyklus `while` opakuje svoje telo, **kým platí podmienka**. Podmienka sa testuje **pred každým** opakovaním; ak neplatí hneď na začiatku, telo sa nevykoná ani raz.

```
n = 1  
while n < 100:  
    print(n, end=" ")  
    n = n * 2  
print()  
print("Prvá mocnina dvoch nad 100:", n)
```

Výstup:

```
1 2 4 8 16 32 64  
Prvá mocnina dvoch nad 100: 128
```

Telo cyklu musí niečo meniť tak, aby podmienka raz prestala platiť – tu sa `n` zdvojnásobuje. Ak by sme riadok `n = n * 2` vynechali, podmienka by platila stále a program by sa nikdy neskončil (**nekonečný cyklus**). Spustený program vtedy zastavíš klávesmi Ctrl + C alebo tlačidlom Stop.

### 2.1 while so vstupom

`while` je ideálny, keď dopredu nevieme, koľko opakovaní bude – napríklad pri čítaní čísel, kým niekto nezadá nulu:

```
sucet = 0  
cislo = int(input("Číslo (0 = koniec): "))  
while cislo != 0:  
    sucet += cislo  
    cislo = int(input("Číslo (0 = koniec): "))  
print("Súčet:", sucet)
```

Výstup:

```
Číslo (0 = koniec): 12  
Číslo (0 = koniec): 7  
Číslo (0 = koniec): 30  
Číslo (0 = koniec): 0  
Súčet: 49
```

Prvé číslo načítame **pred** cyklom, aby mala podmienka čo testovať; ďalšie čísla čítame na konci tela cyklu.

## 2.2 for alebo while?

- **for** – keď vieš, koľkokrát sa má cyklus vykonať, alebo prechádzaš prvky (čísla z `range`, znaky textu, prvky zoznamu).
- **while** – keď počet opakovaní závisí od niečoho, čo sa ukáže až počas behu (kým sa nezadá 0, kým číslo nie je 1, kým nenájde hľadanú hodnotu).

Každý **for** cyklus s `range` sa dá prepísať na **while**:

```
i = 0          # začiatok
while i < 5:    # podmienka
    print(i, end=" ")
    i += 1     # krok
print()
```

Výstup:

```
0 1 2 3 4
```

## 3 Počítadlo a súčet

Veľmi časté úlohy v cykloch sú **spočítať** prvky s nejakou vlastnosťou a **sčítať** hodnoty. Postup je vždy rovnaký:

1. **pred** cyklom premennú vynuluj (`pocet = 0`, `sucet = 0`),
2. **v** cykle ju zväčšuj (`pocet += 1`, `sucet += x`),
3. **po** cykle výsledok použi.

```
sucet = 0
pocet_parnych = 0
for x in range(1, 21):
    sucet += x
    if x % 2 == 0:
        pocet_parnych += 1
print("Súčet 1 až 20:", sucet)
print("Párnych čísel:", pocet_parnych)
```

Výstup:

```
Súčet 1 až 20: 210
Párnych čísel: 10
```

Pri **súčine** sa začína jednotkou, nie nulou (nula by všetko vynulovala). Príklad: faktoriál  $6! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6$ .

```
faktorial = 1
for i in range(1, 7):
    faktorial *= i
print("6! =", faktorial)
```

Výstup:

```
6! = 720
```

### 3.1 Priemer z načítaných čísel

```
n = int(input("Koľko čísel? "))
sucet = 0
for _ in range(n):
    sucet += int(input())
print(f"Priemer: {sucet / n:.2f}")
```

Výstup:

```
Koľko čísel? 5
8
10
7
9
6
Priemer: 8.00
```

### 3.2 Trasovanie cyklu

Pri trasovaní zapisuj hodnoty premenných po každom opakovaní. Program `sucet = 0`, `for i in range(1, 5): sucet += i`:

| opakovanie  | i | sucet po kroku |
|-------------|---|----------------|
| pred cyklom | – | 0              |
| 1.          | 1 | 1              |
| 2.          | 2 | 3              |
| 3.          | 3 | 6              |
| 4.          | 4 | 10             |

## 4 break a continue

- `break` cyklus **okamžite ukončí** – pokračuje sa prvým príkazom za cyklom.
- `continue` **preskočí zvyšok** aktuálneho opakovania a pokračuje ďalším.

```
for i in range(1, 10):
    if i == 6:
        break
    if i % 2 == 0:
        continue
    print(i)
```

Výstup:

```
1
3
5
```

Čísla 2 a 4 sa preskočili (`continue`), pri 6 cyklus skončil (`break`).

## 4.1 while True a break

Častý vzor: cyklus „navždy“, ktorý sa ukončí `break`, keď nastane podmienka. Hodí sa, keď sa podmienka ukončenia dá otestovať až uprostred tela:

```
while True:
    heslo = input("Heslo: ")
    if heslo == "Tajne123":
        break
    print("Nesprávne, skús znova.")
print("Prístup povolený.")
```

Výstup:

```
Heslo: ahoj
Nesprávne, skús znova.
Heslo: heslo
Nesprávne, skús znova.
Heslo: Tajne123
Prístup povolený.
```

## 4.2 Hľadanie s break

```
# prvé číslo väčšie ako 100, ktoré je deliteľné 7 aj 9
n = 101
while True:
    if n % 7 == 0 and n % 9 == 0:
        break
    n += 1
print(n)
```

Výstup:

```
126
```

## 5 Vnorené cykly

V tele cyklu môže byť ďalší cyklus. **Vnútorňý cyklus prebehne celý pri každom opakovaní vonkajšieho.** Ak vonkajší beží 3-krát a vnútorňý 4-krát, telo vnútorného sa vykoná  $3 \cdot 4 = 12$ -krát.

```
for riadok in range(1, 4):
    for stlpec in range(1, 5):
        print(riadok * stlpec, end="\t")
    print()
```

Výstup:

|   |   |   |    |
|---|---|---|----|
| 1 | 2 | 3 | 4  |
| 2 | 4 | 6 | 8  |
| 3 | 6 | 9 | 12 |

Takto vznikla malá tabuľka násobenia: vonkajší cyklus určuje riadok, vnútorňý vypíše stĺpce jedného riadku a `print()` riadok ukončí.

## 5.1 Obrazce zo znakov

```
n = 5
for i in range(1, n + 1):
    print("*" * i)
```

Výstup:

```
*
**
***
****
*****
```

Počet hviezdíčiek v riadku závisí od čísla riadku. Pyramída potrebuje aj medzery pred hviezdíčkami:

```
n = 4
for i in range(1, n + 1):
    print(" " * (n - i) + "*" * (2 * i - 1))
```

Výstup:

```
   *
  **
 ***
****
*****
```

Pri obrazcoch si najprv napíš do tabuľky, koľko medzier a znakov má mať každý riadok, a hľadaj vzťah s číslom riadku `i`.

## 6 Riešené úlohy

### 6.1 Ciferný súčet

**Zadanie:** Načítaj prirodzené číslo a vypíš súčet jeho cifier.

**Riešenie:** Poslednú cifru získame ako `n % 10`, potom ju „odtrhneme“ celočíselným delením `n // 10`. Opakujeme, kým číslo nie je 0 – počet cifier dopredu nepoznáme, preto `while`.

```
n = int(input("Číslo: "))
sucet = 0
while n > 0:
    sucet += n % 10
    n = n // 10
print("Ciferný súčet:", sucet)
```

Výstup:

```
Číslo: 90417
Ciferný súčet: 21
```

| n     | n % 10 | sucet  |
|-------|--------|--------|
| 90417 | 7      | 7      |
| 9041  | 1      | 8      |
| 904   | 4      | 12     |
| 90    | 0      | 12     |
| 9     | 9      | 21     |
| 0     | –      | koniec |

## 6.2 Collatzova postupnosť

**Zadanie:** Začni od zadaného čísla. Ak je párne, vydeľ ho dvoma; ak je nepárne, vynásob ho tromi a pripočítaj 1. Opakuj, kým nedostaneš 1. Vypíš postupnosť a počet krokov.

**Riešenie:** Počet krokov nepoznáme, preto `while n != 1`. Matematici predpokladajú, že postupnosť sa vždy dostane k 1 (Collatzova hypotéza), dokázané to však zatiaľ nie je.

```
n = int(input("Začiatok: "))
kroky = 0
print(n, end=" ")
while n != 1:
    if n % 2 == 0:
        n = n // 2
    else:
        n = 3 * n + 1
        kroky += 1
    print(n, end=" ")
print()
print("Počet krokov:", kroky)
```

Výstup:

```
Začiatok: 6
6 3 10 5 16 8 4 2 1
Počet krokov: 8
```

## 6.3 Prevod do dvojkovej sústavy

**Zadanie:** Načítaj prirodzené číslo a vypíš ho v dvojkovej sústave.

**Riešenie:** Číslo opakovane delíme dvoma a zapisujeme zvyšky. Zvyšky čítané **odzadu** tvoria zápis v dvojkovej sústave, preto každú novú cifru pridáme na **začiatok** textu.

```
n = int(input("Číslo: "))
dvojkovo = ""
while n > 0:
    dvojkovo = str(n % 2) + dvojkovo
    n = n // 2
print(dvojkovo)
```

Výstup:

```
Číslo: 13
1101
```

Kontrola:  $1101_2 = 8 + 4 + 0 + 1 = 13$ . Python to vie aj sám: `bin(13)` je `'0b1101'`.

## 6.4 Fibonacciho postupnosť

**Zadanie:** Vypíš prvých 12 členov Fibonacciho postupnosti: 1, 1, 2, 3, 5, 8, ... (každý ďalší člen je súčtom dvoch predchádzajúcich).

**Riešenie:** Pamätáme si posledné dva členy `a` a `b`. V každom kroku vypíšeme `a` a posunieme sa: nové `a` je staré `b`, nové `b` je súčet. Zápis `a, b = b, a + b` vypočíta pravú stranu celú naraz, takže sa stará hodnota `a` nestratí.

```
a, b = 1, 1
for _ in range(12):
    print(a, end=" ")
    a, b = b, a + b
print()
```

Výstup:

```
1 1 2 3 5 8 13 21 34 55 89 144
```

## 6.5 Hádanie čísla

**Zadanie:** Počítač si myslí číslo od 1 do 100. Ty hádaš a program odpovedá „viac“ alebo „menej“, kým neuhádneš. Na konci vypíše počet pokusov.

**Riešenie:** Náhodné číslo vyberie funkcia `random.randint` (téma Modulárny program). Pre ukážku je nastavené `random.seed`, aby bol príklad zakaždým rovnaký – v hre to nerob. Počet pokusov nepoznáme, preto `while`.

```
import random

tajne = random.randint(1, 100)
pokusy = 0
tip = 0
while tip != tajne:
    tip = int(input("Tvoj tip: "))
    pokusy += 1
    if tip < tajne:
        print("Viac.")
    elif tip > tajne:
        print("Menej.")
print("Uhádnuté na", pokusy, "pokusov!")
```

Výstup:

```
Tvoj tip: 50
Menej.
Tvoj tip: 25
Viac.
Tvoj tip: 37
Viac.
Tvoj tip: 43
Menej.
Tvoj tip: 40
Viac.
Tvoj tip: 42
Uhádnuté na 6 pokusov!
```

## 6.6 Prvočísla do n

**Zadanie:** Vypíš všetky prvočísla menšie ako 50.

**Riešenie:** Prvočíslo je číslo väčšie ako 1, ktoré je deliteľné iba 1 a sebou samým. Pre každé číslo spočítame jeho delitele od 2 po  $n - 1$ ; vnútorný cyklus ukončíme `break`, len čo nájdeme prvého deliteľa.

```
for n in range(2, 50):
    prvocislo = True
    for d in range(2, n):
        if n % d == 0:
            prvocislo = False
            break
    if prvocislo:
        print(n, end=" ")
print()
```

Výstup:

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
```

Premenná `prvocislo` je **príznak** (*flag*): na začiatku predpokladáme, že číslo je prvočíslo, a ak nájdeme deliteľa, príznak zmeníme. Rýchlejší test (stačí skúšať delitele po odmocninu) je v téme Algoritmy.

## 7 Časté chyby

- **Chyba „o jednu“:** `range(1, n)` končí pri  $n - 1$ . Pre čísla od 1 do  $n$  vrátane použijte `range(1, n + 1)`.
- **Nekonečný while:** v tele cyklu sa nemení premenná z podmienky (alebo sa mení zlým smerom).
- **Zlé odsadenie:** príkaz, ktorý patrí za cyklus, je odsadený – vykoná sa pri každom opakovaní. Alebo naopak: príkaz, ktorý má byť v cykle, odsadený nie je.
- **Vynulovanie v cykle:** `sucet = 0` napísané **vo vnútri** cyklu vynuluje súčet pri každom opakovaní. Patrí **pred** cyklus.
- **Zmena riadiacej premennej vo for:** priradenie do `i` v tele cyklu nezmení, ktorá hodnota príde v ďalšom kroku – `for` si ju vezme z `range`.

```
for i in range(3):
    print("i =", i)
    i = 100          # nemá vplyv na ďalší krok
```

Výstup:

```
i = 0
i = 1
i = 2
```

## Zhrnutie

- `for i in range(...):` `range(n)` je 0 až  $n-1$ , `range(od, do)` končí pred `do`, tretí argument je krok (aj záporný).
- `while podmienka:` – opakuje, kým podmienka platí; telo musí podmienku raz „pokažiť“.
- `for` pre známy počet alebo prechádzanie prvkov; `while` pre neznámy počet opakovaní.
- Počítadlo a súčet: vynulovať pred cyklom, zväčšovať v cykle; súčin začína od 1.
- `break` ukončí cyklus, `continue` preskočí zvyšok kroku; `while True + break` je bežný vzor.
- Vnorený cyklus prebehne celý pri každom kroku vonkajšieho.

## Úlohy na precvičenie

---

1. Vypíš násobky čísla 7 menšie ako 100 do jedného riadku.
2. Načítaj  $n$  a vypíš súčet  $1^2 + 2^2 + \dots + n^2$ .
3. Načítavaj čísla, kým nepríde záporné; potom vypíš najväčšie z nich a ich počet.
4. Vypíš celú malú násobilku ( $10 \times 10$ ) zarovnanú do stĺpcov.
5. Načítaj číslo a vypíš počet jeho cifier (bez prevodu na text).
6. Vypíš „obrátenú“ pyramídu z hviezdíčiek s  $n$  riadkami.

---

Úlohy k tejto téme nájdeš na [learnwithmente.com](https://learnwithmente.com) v časti Informatika → Python.