Liepājas valsts tehnikums

**Cikls ar priekšnosacījumu (while) programmēšanas valodā Python**

Mācību materiāls

Darba autors: Maksims Telepnevs

Liepāja 2025

**Ievads mācību materiālam: "Cikls ar priekšnosacījumu programmēšanas valodā Python"**

Ciklā ar nosacījumu **while**, cikls tiek izpildīts, kamēr ir patiess tā noteicošais nosacījums. Tāpēc šo ciklu dažreiz sauc arī par **"kamēr" ciklu**. Bieži vien **while** cikls tiek izmantots situācijās, kad nav iespējams iepriekš paredzēt, cik reizes būs jāizpilda cikla pamatteksts. Ikdienas dzīvē **while** ciklu var sastapt algoritmos, piemēram: "Kamēr redzes laukā ir mašīnas, palikt uz vietas" vai "Kamēr kastē ir detaļas, izņemt detaļu no kastes".

**Kāpēc tas ir svarīgi?**

* **Automātizācija** – atkārto darbības bez manuālas ievades.
* **Dinamiska izpilde** – darbojas, kamēr nosacījums ir izpildīts.
* **Efektivitāte** – samazina koda dublēšanu.

***Cikls while piemers:***Programma, kas izvada skaitļus no 1 līdz 10, izmantojot while ciklu:

# Inicializē mainīgo i ar sākuma vērtību 1

i = 1

# Kamēr i ir mazāks vai vienāds ar 10, izpildīt ciklu

while i <= 10:

# Izvada pašreizējo i vērtību

print(i)

# Palielina i par 1 (sagatavo nākamo vērtību)

i = i + 1

**Detalizēts skaidrojums**

1. **Inicializācija**

i = 1 – Mainīgais i tiek uzstādīts uz sākuma vērtību 1.

1. **Cikla nosacījuma pārbaude**

while i <= 10: – Pārbauda, vai i vērtība nepārsniedz 10.

Ja nosacījums patiess (≤10), izpilda cikla darbības.

Ja nepatiess (>10), cikls pārtrauc darbību.

1. **Cikla darbības**

print(i) – Izvada i pašreizējo vērtību.

i = i + 1 – Palielina i par 1 (piemēram, no 1 uz 2).

1. **Cikla atkārtošanās**

Pēc katras iterācijas programma atgriežas pie nosacījuma pārbaudes:

i = 1 → izpilda (izvada 1, i kļūst 2)

i = 2 → izpilda (izvada 2, i kļūst 3)

...

i = 10 → izpilda (izvada 10, i kļūst 11).

1. **Cikla pārtraukšana**

Kad i = 11:

Nosacījums 11 <= 10 kļūst nepatiess.

Cikls beidz darboties.

Programma turpina ar nākamajām instrukcijām (ja tādas ir).

1. **Rezultāts pēc cikla**

Beigās mainīgais i saglabā vērtību 11, jo tā bija pēdējā piešķirtā vērtība pirms nosacījuma pārbaudes.

***Izvade konsolē:***

1

2

3

4

5

6

7

8

9

10

**Vispārīga sintakse:**

while **nosacījums**:

darbiba\_1

darbiba\_2

...

darbiba\_n

**Obligātās cikla sastāvdaļas:**

1. **Nosacījums**

* Rakstās aiz atslēgvārda **while**
* Jābūt loģiskai izteiksmei (izteiksme, kas atgriež **True/False**)
* Obligāti jāietver kaut viens no:
  + **Salīdzināšanas operatori**: ==, !=, <, >, <=, >=
  + **Loģiskie operatori**: and, or, not kombinācijās
* Pēc nosacījuma vienmēr liek **kolu :**

***Piemērs:***

**while** (x < 10) **and** (y != 0): # Kombinēts nosacījums

1. **Cikla pamatteksts**

* Viena vai vairākas darbības
* Obligāta atkāpe (tradicionāli 4 atstarpes)
* Darbības izpildās secīgi

***Piemērs:***

**while** x **>** 0:

print("Vērtība:", x)

x **-=** 1 # Atkāpe nozīmē, ka šī ir cikla daļa

1. **Mainīgo atjaunināšana**
   * Obligāti jāietver darbības, kas **maina nosacījumā lietotos mainīgos**
   * Bez šīm darbībām risks: **bezgalīgs cikls**
   * Tipiskas formas:
     + Skaitītāja palielināšana: skaitītājs = skaitītājs + 1
     + Vērtības samazināšana: x -= 0.5
     + Statuss maiņa: aktivs = False

**Būtiskie brīdinājumi**

Ja nosacījums nekad nekļūst par **False**, cikls nebeigsies. Tipiskas kļūdas:

1. **Aizmirsta izmaiņa**

x = 10

**while** x **>** 0: # x vienmēr būs > 0

print(x) **# Bez x = x - 1**

1. **Nepareizs nosacījums**

**while True:** **# Vienmēr patiess**

print("∞")

**Pareizs piegājiens**

Vienmēr nodrošināt iespēju **iziet** no cikla:

datu\_skaits = 5

while datu\_skaits > 0:

apstrādāt\_datus()

datu\_skaits -= 1 **# Obligāta samazināšana**

**Cikla vadība ar (break, continue)**

Paskaidrojums**:**

* **break**- pārtrauc ciklu priekšlaicīgi.
* **continue**- izlaiž atlikušo kodu un sāk nākamo iterāciju.

***Piemērs:*** ***Izvade konsolē:***

# Piemērs ar 'break'

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Содержимое, созданное искусственным интеллектом, может быть неверным.i = 1

while True:

if i > 5:

**break** # Pārtrauc ciklu, ja i > 5

print(i)

i += 1

# Piemērs ar 'continue'

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Содержимое, созданное искусственным интеллектом, может быть неверным.i = 0

while i < 10:

i += 1

if i % 2 == 0:

**continue** # Izlaiž pāra skaitļus

print(i)

**while cikls ar else**

Paskaidrojums:

* **else** daļa izpildās, ja cikls nav pārtraukts ar **break**.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.***Piemērs:*** ***Izvade konsolē:***

i = 1

while i <= 5:

print(i)

i += 1

**else**:

print("Cikls veiksmīgi pabeigts!") # Izvadīsies, ja nebūs **'break'**

**Praktiskie uzdevumi: Cikls ar priekšnosacījumu programmēšanas valodā Python**

**1. Uzdevums:** Pāra skaitļu izvade no 2 līdz 20

Izveidojiet programmu, kas izvada visus pāra skaitļus no 2 līdz 20, izmantojot **while** ciklu. Izmantojiet **continue**, lai izlaistu nepāra skaitļus.

**Risinājums:**

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.**

***Izvade konsolē:***

**Изображение выглядит как текст, снимок экрана, Шрифт, черный

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**Paskaidrojums:**

1. Cikls sākas ar **i = 2**.
2. Katrā iterācijā tiek pārbaudīts, vai i ir pāra skaitlis **i % 2 == 0**. Ja nē, tiek izmantots **continue**, lai pārietu uz nākamo iterāciju.
3. Ja skaitlis ir pāra, tas tiek izvadīts, un skaitītājs palielinās par 1.

**2. Uzdevums**

**3. Uzdevums**

**Secinājumi**

**Izmantotajie informācijas avoti:**

1. <https://informatics.msk.ru/mod/book/view.php?id=13732>
2. <https://skillbox.ru/media/code/tsikly-v-python-kak-rabotayut-i-kakie-byvayut/#stk-2>
3. <https://pythontutor.ru/lessons/while/>
4. <https://skolo.lv/mod/book/view.php?id=81332297&chapterid=343085#mod_book-chapter>