Курс програмског језика Пајтон ниво 0

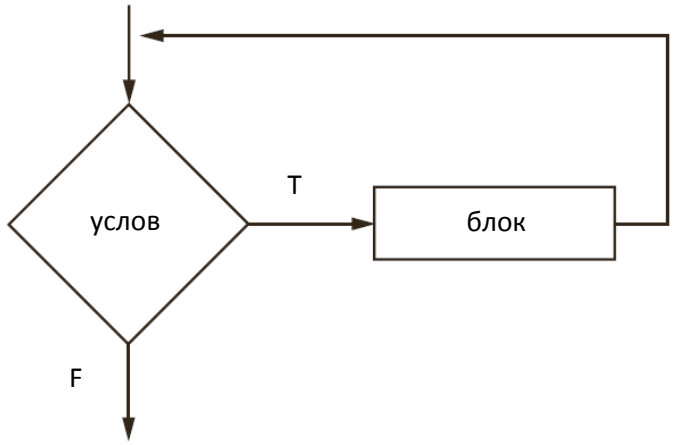
06 час – while петља

1. **Садржај**

2. **Кратак опис**

Ове структуре се користе када је потребно више пута извршити одређене инструкције. Често се структуре понављања називају и петље (loop). Петље се деле на две категорије, петље контролисане условом (condition-cntrolled) и петље контролисане бројем понављања (count-controlled).

3. **While петља**



Ово је петља контролисана условом јер се број понављања контролише тачно/нетачно условом. Све док је услов испуњен блок наредби унутар while петље ће се понављати.

У случају да је услов тачан, извршава се блок наредби и поново се испитује услов.

Ако услов није тачан излази се из петље.

Пример 01:

```
# program izracunava zaradu trgovca od novca dobijenog transakcijom

jos = 'da'    # kreiranje promenjive za kontrolu petlje
while jos == 'da':
    novac = float(input('Uneti novac dobijen u transakciji: '))
    procenat_zarade = float(input('Uneti procenat zarade trgovca: '))

    zarada = novac * procenat_zarade

    print('Zarada je ', zarada, ' dinara.')
    print('Da li treba uneti jos transakcija? ')
    jos = input('(Uneti da za dalji rad): ')
```

```
Uneti novac dobijen u transakciji: 50000
Uneti procenat zarade trgovca: 0.10
Zarada je  5000.0  dinara.
Da li treba uneti jos transakcija?
(Uneti da za dalji rad): da
```

```
Uneti novac dobijen u transakciji: 300000
Uneti procenat zarade trgovca: 0.15
Zarada je  45000.0  dinara.
Da li treba uneti jos transakcija?
(Uneti da za dalji rad):
>>>
```

Промењива **jos** добија вредност 'da' и испитивање вредности ове промењиве чини услов по којем ради петља. Програмер мора обезбедити да се може променити вредност промењиве **jos** унутар тела петље, јер само тако се може изаћи из петље.

НАПОМЕНА

While петља се назива и претест петља, што значи да она тестира услов пре него се изведе итерација (један циклус обраде података у петљи). Пошто се тестирање услова изводи пре почетка петље најчешће је потребно урадити неке кораке пре извршења петље да би били сигурни да ће се петља извести макар једном. А то значи да мора постојати посебна промењива и да мора имати одговарајућу вредност. Само на тај начин смо сигурни да ће се петља извршити макар једном. Такође, могући су случајеви када услов није испуњен пре првог уласка у петљу, али су то специфичне и ретке ситуације.

*Посебан случај петље је бесконачна петља (infinite loops). У пракси, циљ програмера је да ниједна петља не постане бесконачна петља. Ако је петља бесконачна то значи да није омогућила да испитивани услов постане нетачан и зато се никада неће завршити њено извршавање (тј све до прекида рада програма).

```
ciklus = 10
while ciklus > 1:
        print(“Zdravo, ja sam beskonacna petlja.“)
```

*Стражари (sentinels) су посебне вредности које означавају крај секвенце вредности. Користе се када програмер не зна унапред које ће вредности бити у секвенци петље али се зна да ће се користити велики број вредности. Када програм учита вредност стражара, зна да је дошао до краја секвенце, па се петља завршава. Зато вредност стражара мора бити јединствена и не сме се заменити за неку другу.

Пример 02:

```
# Program racuna porez na nekretninu

porez_faktor = 0.0065

print('Uneti bilo koju vrednost za dalji rad ili uneti 0 za zavrsetak rada.')
straza = int(input('Vrednost strazara: '))

while straza != 0:
    vrednost_nekretnine = float(input("Vrednost nekretnine: "))
    porez = vrednost_nekretnine * porez_faktor
    print("Porez na nekretnine: ", porez)
    print('Uneti bilo koju vrednost za dalji rad ili uneti 0 za zavrsetak rada.')
    straza = int(input('Vrednost strazara: '))
```

```
Uneti bilo koju vrednost za dalji rad ili uneti 0 za zavrsetak rada.
Vrednost strazara: 100
Vrednost nekretnine: 1000000
Porez na nekretnine:  6500.0
Uneti bilo koju vrednost za dalji rad ili uneti 0 za zavrsetak rada.
Vrednost strazara: 200
Vrednost nekretnine: 1500000
Porez na nekretnine:  9750.0
Uneti bilo koju vrednost za dalji rad ili uneti 0 za zavrsetak rada.
Vrednost strazara: 0
>>>
```

4. **Петља за валидацију улаза (ПВУ)**

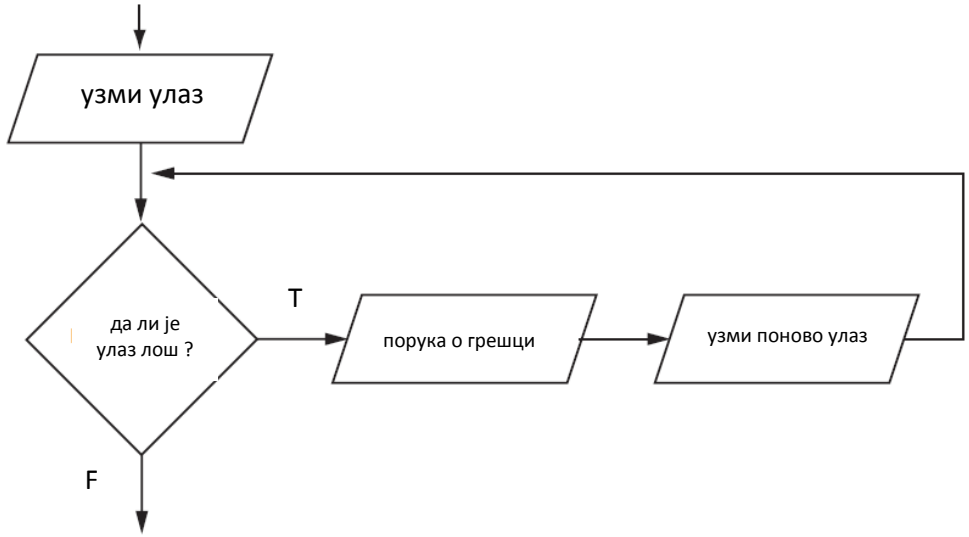
ПВУ је процес провере података који су унешени у програм да би се осигурало да су валидни пре него што ће се користити у обради података.

Обично се валидација улаза изводи у петљи која итерира све док улазна промењива упућује на лоше податке.

У програмирању постоји изрека “garbage in, garbage out“ (GIGO), што значи да лоши улазни подаци стварају лоше излазне резултате. Пошто рачунар не распознаје сам по себи шта је од улазних података добар а шта лош податак, мора постојати код којим се постиже провера улаза.

Најчешће коришћена техника ПВУ:

У овој методи учита се улаз па се стартује петља. Ако је улаз лош, петља ерализује блок наредби. Петља прикаже поруку о грешци тако да корисник зна да је улаз лош а онда се учита нов улаз. Петља се понавља све док је улаз лош.



Интересантан детаљ је да се улаз уноси у коду на два места, на почетку структуре (примарно читање) и унутар петље.

Пример 03:

```
rezultat_testa = int(input('Uneti rezultat testa: '))
while rezultat_testa < 0:
    print('GRESKA: rezultat testa ne moze biti negativan!')
    rezultat_testa = int(input('Uneti rezultat testa: '))
```

```
Uneti rezultat testa: 0
>>>
========== RESTART: C:/Users/nera/Documents/pa
Uneti rezultat testa: 100
>>>
========== RESTART: C:/Users/nera/Documents/pa
Uneti rezultat testa: -1
GRESKA: rezultat testa ne moze biti negativan!
Uneti rezultat testa: 3
>>>
```

Често се у ПВУ сама петља назива замка за грешке (error trap) или обрађивач грешке (error handler).

5. **Увучене петље**

Увучена петља се назива петља која се налази унутар тела друге петље.

Пример 04:

```
a = 1
while a > 0:
    b = int(input("Uneti neku vrednost: "))
    while b != 0:
        print(b)
        b = 0
    a = int(input("Idemo dalje? "))
```

```
Uneti neku vrednost: 3
3
Idemo dalje? 1
Uneti neku vrednost: 3
3
Idemo dalje? 5
Uneti neku vrednost: 0
Idemo dalje? 0
>>>
```

У примеру се користе две промењиве: **a** која је стражар за спољну петљу и **b** која је промењива везана за унутрашњу петљу.

Спољна петља ће се најмање једном извршавати, пошто је иницијална вредност стражара различита од 0. Спољна петља ће се извршавати све док стражар не добије вредност једнаку 0, а то ће зависити од жеље корисника (Idemo dalje?), што изазива и излазак из програма.

Унутрашња петља одмах зависи од жеље корисника па је могуће да се ниједном не изврши (ако корисник одмах унесе **b** = 0).

Тест

1. Петља контролисана _____ користи тачно/нетачно услове за контролу броја понављања.
   а) Булом
   б) условом
   в) одлуком
   г) бројањем
2. Свако понављање петље се зове:
   а) услов
   б) револуција
   в) орбита
   г) итерација
3. While петља је _____ врста петље.
   а) претест
   б) не-тест
   в) предквалификације
   г) пост-итерација
4. Пре првог извршавања while петље врши се _____.
   а) тестирање услова
   б) провера итерација
   в) промена вредности промењиве
   г) бројање циклуса

5. _____ петља нема начина да се заврши и понавља се све док се програм не прекине.
   а) недертеминисана
   б) непрекидна
   в) бесконачна
   г) безвремена
6. Шта су то стражари ?
7. На који начин се стражар има везе са петљом ?
8. Шта је то петља за валидацију улаза ?
9. Петља за валидацију улаза се налази у коду:
   а) испред главне петље
   б) на почетку главне петље
   в) после обраде података у главној петљи
   г) после главне петље
10. Зашто је неопходно коришћење ПВУ ?
11. Који део структуре се два пута понавља коришћењем ПВУ ?
12. На шта се односе називи error trap и error handler ?
    а) промењиве за исправљање грешака
    б) различите процесе унутар ПВУ
    в) претраживање тачних података по петљи
    г) петље унутар ПВУ
13. While петље не могу бити увучене петље (да/не) ?
14. Шта су то увучене петље ?
15. Увучене петље могу имати исте промењиве као део услова ?
    а) да, увек
    б) не
    в) да, под одређеним условима

Вежбања

1. а) Шта ће се појавити на екрану по реализацији кода ?
   num 1 = 10
   num 2 = 5
   num 1 = num 1 + num 2
   print(num1)
   print(num2)
   б) Исправити код из вежбања 1, да би приказао жељене податке.
2. Написати код за једну бесконачну петљу. Урадити минимум измена у коду да би се исправио такав код.
3. Напиши while петљу која тражи од корисника број. Број треба помножити са 10, а резултат додели промењивој **product**. Петља треба да итерира све док је **product** мањи од 100.
4. Напиши програм који прима два броја од корисника, проверава ако је делилац 0, дели их и приказује њихов количник. Ако је делилац 0, упозорава корисника са “Дељење није могуће“.

5. Шта ће се исписати на екрану после следећег кода ?

```
i = 1
j = 8
while i < j:
        print(i + j)
        i ++
        j –
```

6. Написати код користећи while петље, који исписује сва велика слова абецеде тако да се свако слово испише онолико пута колико износи његов редни број у абецеди.

```
A                                                    chr(65) = A, chr(66) = B…
BB
CCC
DDDD
EEEEEE
....
```

7. Који су природни бројеви од 1 до 100 који су дељиви и са 4 и са 7 ?
8. За сваки учитани негативан број исписати на екрану његову апсолутну вредност.

Задаци

1. **Сума бројева**
   Написати програм са петљом који тражи од корисника да унесе серију позитивних бројева. Корисник треба да унесе негативан број као сигнал за крај серије. После уноса свих позитивних бројева, програм приказује њихов збир.
2. **Природни бројеви у опсегу**
   Исписати редом природне бројеве од 5 до 10.
3. **Збир n природних бројева**
   Корисник уноси природни број n а код исписује збир првих n природних бројева.
4. **Нивои океана**
   Претпостављајући да се нивои океана повећавају за 1.6 милиметара годишње, направити апликацију која приказује број милиметара за које ће се океан подићи сваке године за следећих 25 година.
5. **Парни до n**
   Исписати на екрану све парне природне бројеве од 1 до n.
6. **Непарни дељиви са 3**
   Написати производ првих n непарних природних бројева дељивих са 3.
7. **Збир природних у опсегу**
   Приказати збир природних бројева од а до б.
8. **Збир и бројност**
   Приказати на екрану збир парних бројева до броја **n**, дељивих са 7 и њихов укупан број.
9. **Математика све до 0**
   Учитане бројеве сабирати ако су позитивни а ако су негативни одузимати. Када се учита 0 прекида се програм и исписује коначан резулата