Низови

Секвенца (низ) је објекат који садржи велики број ствари који су заправо подаци.
Ствари у низу се смештају један после другог.
Пајтон омогућава велики број начина на које се изводе операције на стварима у низу.
Постоји неколико различитих типова објеката низова у Пајтону.
Два типа низова су основна: nторке и листе, обе су секвенце које могу садржавати различите типове података.

nторке

Nторке су врста секвенци (низова) и оне су непромењиве (immutable).
То значи да када се једном направе, не могу се променити.
Када се креира nторка, њени елементи се затварају у заграде:

```
>>> moja_ntorka = (1, 2, 3, 4, 5)
>>> print(moja_ntorka)
(1, 2, 3, 4, 5)
>>>
```

У првој линији се креира нторка додељивањем листе елемената промењивој moja_ntorka.

Увод у листе

Листа је објекат који садржи више ствари који су заправо подаци.
Свака од ствари смештена у листи је елемент.
Исказ који креира листу целих бројева: parni_brojevi = [2, 4, 6, 8, 10]
Ствари у угластим заградама и раздвојени зарезима су елементи листе.
Када се изврши претходни исказ, промењива parni_brojevi ће указивати на листу.
Елементи у листи не морају да буду бројеви: imena = [‘Miki’, ‘Kiki’, ‘Jovan’, ‘Ana’, ‘Dragana’].
Такође, листа може садржати елементе различитих типова података: info = [‘Dejan’, 30, 34.589].
Функција print приказује листу елемената: print(info), даје: [‘Dejan’, 30, 34.589]
Пајтон има уграђену функцију list() која конвертује неке типове објеката у листе.
Нпр, пошто је range функција која враћа итерабилни податак, а то је објекат који садржи серију вредности преко које се може итерирати.
Може се користити исказ попут: brojevi = list(range(5)).
После извршења исказа дешава се следеће:

- Функција range се позива са 5 достављеним као аргументом, функција враћа итерабилне вредности 0, 1, 2, 3, 4
- Итерабилна вредност је пренешена као аргумент у функцију list(), функција list() враћа листу [0, 1, 2, 3, 4]
- lista[0, 1, 2, 3, 4] је додељена brojevi промењивој

broj = list(range(1, 10, 2))
Када се придодају три аргумента у функцију range, први аргумент је почетна вредност, други лимитира низ а трећи је вредност корака.
Зато ће листа broj имати следеће елементе: [1, 3, 5, 7, 9].

Итерација по листи са for петљом

Познате су технике приступа појединачном знаку унутар стринга.
Многе од тих техника се могу применити и у листама.
Нпр, може се итерирати по листи са for петљом:

```
broj = [99, 100, 101, 102]
for n in broj:
    print(n)
```

Индексирање

Други начин да се приђе појединачном елементу у листи је са индексом.
Сваки елемент у листи има индекс који одређује позицију елемента у листи.
Индекси започињу са 0, први елемент има индекс 0, други 1, итд.
Последњи елемент у листи од n елемената има индекс n -1.
moja_lista = [10, 20, 30, 40] има 4 елемента у листи и њихови индекси су 0, 1, 2 и 3.
Елементи ове листе се могу одштампати на овај начин:
print(moja_lista[0], moja_lista[1], moja_lista[2], moja_lista[3])

Пример: штампање елемената листе помоћу петље

```
moja_lista = [10, 20, 30, 40]
indeks = 0
while indeks < 4:
    print (moja_lista[indeks])
    indeks += 1
```

Такође се могу користити негативни индекси за идентификовање позиције елемента релативно у односу на крај листе.

Пајтон интерпретер додаје негативне индексе на дужину листе за одређивање позиције елемента.

Индекс -1 идентификује последњи елемент у листи, -2 идентификује следећи до њега, итд.

```
>>> moja_lista = [10, 20, 30, 40]
>>> print(moja_lista[-1], moja_lista[-2], moja_lista[-3], moja_lista[-4])
40 30 20 10
```

Ако се напише погрешан индекс за листу, нпр у претходном случају : print(moja_lista[4]), појавиће се објава о грешци а то се за овај случај назива изузетак о грешци индекса (IndexError exception).

Функција len

Пајтон има уграђену функцију len која враћа дужину низа, попут листа.

```
>>> moja_lista = [10, 20, 30, 40]
>>> velicina = len(moja_lista)
>>> print(velicina)
4
```

Прва линија кода додељује листу промењивој moja_lista.

Друга линија кода позива функцију len, придодајући промењиву moja_lista као аргумент.

Функција враћа вредност 4, што је број елемената у листи.

Функција len се може користити за спречавање IndexError изузетка при итерацији по листи са петљом.

Пример

```
moja_lista = [10, 20, 30, 40]
indeks = 0
while indeks < len(moja_lista):
    print (moja_lista[indeks])
    indeks += 1
```

Промењивост листа

Листе у Пајтону су промењиве (mutable), што значи да се њихови елементи могу мењати.

То значи да се форма list[indeks] може појавити на левој страни оператора доделе.

Пример 2)

```
>>> brojevi = [1, 2, 3]
>>> print(brojevi)
[1, 2, 3]
>>> brojevi[0] = 99
>>> print(brojevi)
[99, 2, 3]
```

Када се користи индексирани израз за доделу вредности елементу листе, мора се користити дозвољен индекс за постојећи елемент или ће се појавити IndexError изузетак.

```
>>> brojevi = [1, 2, 3]
>>> brojevi[5] = 100
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list assignment index out of range
```

Пошто у првој линији де дефинисано да листа има 3 елемента, са индексима од 0 до 2, покушај мењања непостојећег елемента у низу изазива прекид програма и подизање изузетка.

Да би се користио индексирани израз за попуњавање листе вредности, прво се мора направити листа:

Надовезивање листа

Надовезивање (concatenate) значи удруживање два ствари и за то се користи + оператор.

```
>>> list1 = [1, 2, 3, 4]
>>> list2 = [5, 6, 7, 8]
```

```
>>> imena_devojcice = ['Ana', 'Dragana', 'Gordana']
>>> imena_decaci = ['Milan', 'Dejan', 'Zoran']
```

```
>>> list3 = list1 + list2
>>> print(list3)
[1, 2, 3, 4, 5, 6, 7, 8]
```

```
>>> imena = imena_devojcice + imena_decaci
>>> print(imena)
['Ana', 'Dragana', 'Gordana', 'Milan', 'Dejan', 'Zoran']
```

Такође, може се користити појачани оператор доделе:

```
>>> list1 = [1, 2, 3, 4]
>>> list2 = [5, 6, 7, 8]
>>> list1 += list2
>>> print(list1)
[1, 2, 3, 4, 5, 6, 7, 8]
```

```
>>> imena = ['Ana', 'Zora', 'Miki']
>>> imena += ['Zoki', 'Darko']
>>> print(imena)
['Ana', 'Zora', 'Miki', 'Zoki', 'Darko']
```

Одсецање листа

Понекад је потребно изабрати више од једног елемента из секвенце.

У Пајтону, могу се писати изрази који бирају делове секвенце, познате као одсечци.

Одсечак (slice) је неки број ствари које су заједно одстрањене из секвенце.

Када се узме одсечак из листе, добија се неки број елемената из листе:

naziv_liste[start : kraj]

Где је start индекс првог елемента у одсечку, а kraj је индекс који ограничава крај одсечка.

Зато прошли пример враћа листу која садржи копију елемената од индекса start до (не укључујући) елемент са индексом kraj.

```
>>> dani = ['pon', 'uto', 'sre', 'cet', 'pet', 'sub', 'ned']
>>> sredina = dani[2:5]
>>> print(sredina)
['sre', 'cet', 'pet']
```

Види се да исказ `dani[2:5]` изводи одсецање из оригиналне листе од индекса 2 до индекса 4 чиме се креира нова листа која се зове sredina.

Ако се изостави индекс start у изразу за одсецање, Пајтон користи 0 као почетни индекс по дифолту:

```
>>> brojevi = [1, 2, 3, 4, 5]
>>> print(brojevi[:3])
[1, 2, 3]
>>> print(brojevi)
[1, 2, 3, 4, 5]
```

Види се да одсецањем дела оригиналне листе се не мења оригинална листа јер одсецањем се прави копија одсечка оригиналне листе.

Ако се изостави индекс краја одсечка, по дифолту се иде до краја листе:

```
>>> brojevi = [1, 2, 3, 4, 5]
>>> print(brojevi[3:])
[4, 5]
```

Ако се изоставе оба индекса у одсечку, добија се копија целе листе:

```
>>> brojevi = [1, 2, 3, 4, 5]
>>> print(brojevi[:])
[1, 2, 3, 4, 5]
```

Изрази за одсецање такође могу имати и вредност корака, која омогућава прескакање неких елемената у листи:

```
>>> brojevi = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> print(brojevi)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> print(brojevi[1:8:2])
[2, 4, 6, 8]
```

Трећи број у примеру је корак одсецања и он упућује да се у одсечак укључује сваки други елемент почевши од задатог индекса.

Могуће је користити и негативне индексе у одсецању при чему Пајтон додаје негативан индекс на дужину листе да би се добила позиција на коју упућује индекс:

```
>>> brojevi = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> print(brojevi[-5:])
[6, 7, 8, 9, 10]
```

Погрешни индекси не доводе до појаве изузетака:

- Ако kraj индекс указује на позицију изван краја листе, Пајтон ће употребити дужину листе уместо тога

- Ако start индекс специфицира позицију пре почетка листе, Пајтон ће аутоматски употребити 0
- Ако је start индекс већи од kraj индекса, одсецање ће вратити празну листу

Лист методе

Листе имају на располагању велики број метода које омогућавају додавање елемената, одстрањивање елемената, промену редоследа елемената итд.

1. Метода append

Ова метода се користи за додавање ствари у листу.
Ствар која се придодаје као аргумент је придружена као последњи елемент у листи.

```
lista = []
ponovo = 'da'
while ponovo == 'da':
    ime = input('Unesi ime: ')
    lista.append(ime)
    print('Da li treba dodati jos jedno ime?')
    ponovo = input('da ili bilo sta drugo za ne: ')
    print()
print('Ovo su unesena imena:')
for ime in lista:
    print(ime)
```

2. Метод index

Ако је потребно знати на којој се позицији налази ствар у листи, користи се овај метод.
Придодаје се аргумент у методу index а враћа се индекс првог елемента у листи који садржу ту ствар која се тражи.
Ако ствар није пронађена у листи, метод подиже изузетак ValueError.

```
hrana = ['pica', 'keks', 'sladoled']
stvar = input('Koja te hrana interesuje? ')
stvar_indeks = hrana.index(stvar)
print('Pozicija', stvar, 'u listi je', stvar_indeks)
```

```
Koja te hrana interesuje? keks
Pozicija keks u listi je 1
```

3. Метод insert

Овај метод омогућава уметање ствари у листу на жељену позицију.
Придодају се два аргумента методи insert: индекс где та ствар треба да се смести у листи и ствар која се жели уметнути у листу.

```
imena = ['Miki', 'Kata', 'Bili']
print('Lista pre umetanja:')
print(imena)
imena.insert(0, 'Koki')
print('Lista posle umetanja:')
print(imena)
```

```
Lista pre umetanja:
['Miki', 'Kata', 'Bili']
Lista posle umetanja:
['Koki', 'Miki', 'Kata', 'Bili']
```

4. Метод sort

Овај метод преуређује елементе листе тако да се они појаве у растућем редоследу (од најниже ка највишој вредности).

```
moja_lista = [9, 1, 0, 2, 8, 6, 7, 4, 5, 3]
print('Originalni redosled:', moja_lista)
moja_lista.sort()
print('Uredjen redosled:', moja_lista)
```

```
Originalni redosled: [9, 1, 0, 2, 8, 6, 7, 4, 5, 3]
Uredjen redosled: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
moja_lista = ['alfa', 'gama', 'delta', 'beta']
print('Originalni redosled:', moja_lista)
```

```
moja_lista.sort()
print('Uredjen redosled:', moja_lista)
```

```
Originalni redosled: ['alfa', 'gama', 'delta', 'beta']
Uredjen redosled: ['alfa', 'beta', 'delta', 'gama']
```

Методе листе

1. Метода remove

Ова метода отклања ствар из листе.
У методи се ствар појављује као аргумент, и први елемент који има ту ствар се отклања из листе.
Овиме се смањује величина листе за један.
Све остале ствари се померају за једну позицију према почетку листе.
Ако ствар не постоји у листи, појављује се изузетак ValueError.

```
hrana = ['pica', 'keks', 'sladoled']
stvar = input('Koju hranu da otkacim iz liste? ')
hrana.remove(stvar)
print('Ovo je promenjena lista: ')
print(hrana)
```

```
Koju hranu da otkacim iz liste? keks
Ovo je promenjena lista:
['pica', 'sladoled']
```

2. Метода reverse

Ова метода обрће редослед ствари у листи:

```
moja_lista = [1, 2, 3, 4]
print('Originalni redosled:', moja_lista)
moja_lista.reverse()
print('Obrnut redosled:', moja_lista)
```

```
Originalni redosled: [1, 2, 3, 4]
Obrnut redosled: [4, 3, 2, 1]
```

Исказ del

У неким ситуацијама је потребно отклонити елемент на одређеном индексу, без обзира која ствар се налази на том индексу.
То се може остварити коришћењем исказа del:

```
moja_lista = [1, 2, 3, 4, 5]
print('Pre brisanja:', moja_lista)
del moja_lista[2]
print('Posle brisanja:', moja_lista)
```

```
Pre brisanja: [1, 2, 3, 4, 5]
Posle brisanja: [1, 2, 4, 5]
```

Функције min и max

У Пајтону се две функције min и max користе за рад са секвенцама.
Функција min прихвата секвенце (попут листа) као аргументе и враћа ствар која има најмању вредност у секвенци.
Функција max прихвата секвенце (попут листа) као аргументе и враћа ствар која има највећу вредност у секвенци.

```
moja_lista = [5, 4, 3, 2, 50, 40, 30]
print('Najnizu vrednost ima', min(moja_lista))
print('Najvecu vrednost ima', max(moja_lista))
```

```
Najnizu vrednost ima 2
Najvecu vrednost ima 50
```

Копирање листа

У Пајтону, додела једне промењиве другој промењивој чини да обе упућују на исти објекат у меморији:

```
>>> list1 = [1, 2, 3, 4]
>>> list2 = list1
>>> print(list1)
[1, 2, 3, 4]
>>> print(list2)
[1, 2, 3, 4]
>>> list1[0] = 99
>>> print(list1)
[99, 2, 3, 4]
```

```
>>> print(list2)
[99, 2, 3, 4]
```

Сада је потребно направити копију листе тако да list1 и list2 упућују на две различите али по садржају исте листе. Један начин да се то оствари је са петљом која копира сваки елемент листе.

```
list1 = [1, 2, 3, 4]
list2 = []
for item in list1:
    list2.append(item)
```

Једноставнији и ефикаснији начин да се исто оствари је употреба оператора надовезивања:

```
list1 = [1, 2, 3, 4]
list2 = [] + list1
```

Сада list1 и list2 указују на две одвојене али идентичне листе.