Innehåll

[Listor, Tuplets och Set 2](#_Toc87682948)

[Listor 2](#_Toc87682949)

[Importer 2](#_Toc87682950)

[Random 2](#_Toc87682951)

[Heltal 2](#_Toc87682952)

# Listor, Tuplets och Set

## Listor

Här är några andra vanliga listmetoder.

* list.append(elem) -- lägger till ett enda element i slutet av listan. Vanligt fel: returnerar inte den nya listan, ändrar bara originalet.
* list.insert(index, elem) -- infogar elementet vid det givna indexet, flyttar element åt höger.
* list.extend(list2) lägger till elementen i list2 i slutet av listan. Att använda + eller += på en lista liknar att använda extend().
* list.index(elem) -- söker efter det givna elementet från början av listan och returnerar dess index. Kastar ett ValueError om elementet inte visas (använd "in" för att kontrollera utan ett ValueError).
* list.remove(elem) -- söker efter den första instansen av det givna elementet och tar bort det (kastar ValueError om det inte finns)
* list.sort() -- sorterar listan på plats (returerar den inte). (Funktionen sortered() som visas senare är att föredra.)
* list.reverse() -- vänder listan på plats (returerar den inte)
* list.pop(index) -- tar bort och returnerar elementet vid det givna indexet. Returnerar elementet längst till höger om index utelämnas (ungefär motsatsen till append()).

Lägg märke till att dessa är \*metoder\* på ett listobjekt, medan len() är en funktion som tar listan (eller strängen eller vad som helst) som ett argument.

# Importer

## Random

### Heltal

Funktioner för heltal

random.**randrange**( *sluta* )

random.**randrange**( *starta* , *stopp*[ , *steg*] )

Returnera ett slumpmässigt valt element från . Detta motsvarar , men bygger faktiskt inte ett intervallobjekt.range(start, stop, step)choice(range(start, stop, step))

Det positionella argumentmönstret matchar det för [range()](https://docs.python.org/3/library/stdtypes.html" \l "range" \o "räckvidd). Nyckelordsargument bör inte användas eftersom funktionen kan använda dem på oväntade sätt.

*Ändrad i version 3.2:*[randrange()](https://docs.python.org/3/library/random.html?highlight=random#random.randrange) är mer sofistikerad när det gäller att producera jämnt fördelade värden. Tidigare använde den en stil som kunde ge lite ojämna fördelningar.int(random()\*n)

*Utfasad sedan version 3.10:* Den automatiska konverteringen av icke-heltalstyper till ekvivalenta heltal är utfasad. För närvarande randrange(10.0)konverteras förlustfritt till randrange(10). I framtiden kommer detta att höja en [TypeError](https://docs.python.org/3/library/exceptions.html" \l "TypeError" \o "Skrivfel).

*Utfasad sedan version 3.10:* Undantaget för icke-integralvärden som randrange(10.5) eller randrange('10')kommer att ändras från [ValueError](https://docs.python.org/3/library/exceptions.html" \l "ValueError" \o "ValueError)till [TypeError](https://docs.python.org/3/library/exceptions.html" \l "TypeError" \o "Skrivfel).

random.**randint**( *a* , *b* )

Returnera ett slumpmässigt heltal *N* så att . Alias ​​för .a <= N <= brandrange(a, b+1)

random.**getrandbits**( *k* )

Returnerar ett icke-negativt Python-heltal med *k*slumpmässiga bitar. Denna metod levereras med MersenneTwister-generatorn och vissa andra generatorer kan också tillhandahålla den som en valfri del av API:et. När tillgängligt, gör det [getrandbits()](https://docs.python.org/3/library/random.html?highlight=random#random.getrandbits)möjligt [randrange()](https://docs.python.org/3/library/random.html?highlight=random" \l "random.randrange" \o "random.randrange)att hantera godtyckligt stora intervall.

Sekvenser

Funktioner för sekvenser

random.**choice**( *seq* )

Returnera ett slumpmässigt element från den icke-tomma sekvensen *seq* . Om *seq* är tom, höjer [IndexError](https://docs.python.org/3/library/exceptions.html" \l "IndexError" \o "IndexError).

random.**choices**( *population* , *vikter = Ingen* , *\** , *cum\_weights = Ingen* , *k = 1* )

Returnera en *k-* stor lista med element valda från *populationen* med ersättning. Om *befolkningen* är tom, höjer [IndexError](https://docs.python.org/3/library/exceptions.html" \l "IndexError" \o "IndexError).

Om en *viktsekvens* anges, görs val enligt de relativa vikterna. Alternativt, om en *cum\_weights-* sekvens ges, görs valen enligt de kumulativa vikterna (kanske beräknat med [itertools.accumulate()](https://docs.python.org/3/library/itertools.html" \l "itertools.accumulate" \o "itertools.ackumulera)). Till exempel är de relativa vikterna ekvivalenta med de kumulativa vikterna . Internt omvandlas de relativa vikterna till kumulativa vikter innan du gör val, så att ange de kumulativa vikterna sparar arbete.[10, 5, 30, 5][10, 15, 45, 50]

Om varken *vikter* eller *cum\_weights* anges, görs val med lika stor sannolikhet. Om en viktsekvens tillhandahålls måste den vara lika lång som *populationssekvensen* . Det är en [TypeError](https://docs.python.org/3/library/exceptions.html" \l "TypeError" \o "Skrivfel) att ange både *vikter* och *cum\_weights* .

De *vikter* eller *cum\_weights* kan använda någon numerisk typ som fungerar tillsammans med de [float](https://docs.python.org/3/library/functions.html#float)värden som returneras av (som inkluderar heltal, flottar, och fraktioner men utesluter decimaler). Vikter antas vara icke-negativa och ändliga. A höjs om alla vikter är noll.[random()](https://docs.python.org/3/library/random.html?highlight=random#module-random)[ValueError](https://docs.python.org/3/library/exceptions.html#ValueError)

För ett givet frö ger [choices()](https://docs.python.org/3/library/random.html?highlight=random" \l "random.choices" \o "slumpmässiga.val)funktionen med lika viktning vanligtvis en annan sekvens än upprepade anrop till [choice()](https://docs.python.org/3/library/random.html?highlight=random#random.choice). Algoritmen som används av [choices()](https://docs.python.org/3/library/random.html?highlight=random" \l "random.choices" \o "slumpmässiga.val)använder flyttalsaritmetik för intern konsistens och hastighet. [choice()](https://docs.python.org/3/library/random.html?highlight=random#random.choice)Algoritmen som används som standard till heltalsaritmetik med upprepade val för att undvika små fördomar från avrundningsfel.

*Nytt i version 3.6.*

*Ändrad i version 3.9:* Höjer a [ValueError](https://docs.python.org/3/library/exceptions.html" \l "ValueError" \o "ValueError)om alla vikter är noll.

random.**shuffle**( *x*[ , *slumpmässigt* ] )

Blanda sekvensen *x* på plats.

Det valfria argumentet *random* är en 0-argumentfunktion som returnerar ett slumpmässigt flytande i [0.0, 1.0); som standard är detta funktionen .[random()](https://docs.python.org/3/library/random.html?highlight=random" \l "random.random" \o "random.random)

För att blanda en oföränderlig sekvens och returnera en ny blandad lista, använd istället.sample(x, k=len(x))

Observera att även för små len(x), kan det totala antalet permutationer av *x* snabbt växa sig större än perioden för de flesta slumptalsgeneratorer . Detta innebär att de flesta permutationer av en lång sekvens aldrig kan genereras. Till exempel är en sekvens med längden 2080 den största som kan passa inom perioden för Mersenne Twisters slumptalsgenerator.

*Utfasad sedan version 3.9, kommer att tas bort i version 3.11:* Den valfria parametern *random* .

random.**sample**( *population* , *k* , *\** , *counts = None* )

Returnera en *k* längd lista med unika element valda från populationssekvensen eller mängden. Används för slumpmässig provtagning utan ersättning.

Returnerar en ny lista som innehåller element från populationen samtidigt som den ursprungliga populationen lämnas oförändrad. Den resulterande listan är i urvalsordning så att alla delsegment också kommer att vara giltiga slumpmässiga urval. Detta gör det möjligt för lotterivinnare (provet) att delas upp i storpris och andra platsvinnare (underslices).

Medlemmar av befolkningen behöver inte vara [hashbara](https://docs.python.org/3/glossary.html" \l "term-hashable) eller unika. Om populationen innehåller upprepningar är varje förekomst ett möjligt urval i urvalet.

Upprepade element kan anges en i taget eller med valfria sökord endast *räknar* parameter. Till exempel motsvarar .sample(['red', 'blue'], counts=[4, 2], k=5)sample(['red', 'red', 'red', 'red', 'blue', 'blue'], k=5)

För att välja ett urval från ett intervall av heltal, använd ett [range()](https://docs.python.org/3/library/stdtypes.html" \l "range" \o "räckvidd)objekt som ett argument. Detta är särskilt snabbt och utrymmeseffektivt för provtagning från en stor population: .sample(range(10000000), k=60)

Om urvalsstorleken är större än populationsstorleken [ValueError](https://docs.python.org/3/library/exceptions.html" \l "ValueError" \o "ValueError) höjs a.

*Ändrad i version 3.9:* Lade till parametern *counts* .

*Utfasad sedan version 3.9:* I framtiden måste *populationen* vara en sekvens. Förekomster av [set](https://docs.python.org/3/library/stdtypes.html" \l "set" \o "uppsättning)stöds inte längre. Uppsättningen måste först omvandlas till a [list](https://docs.python.org/3/library/stdtypes.html" \l "list" \o "lista)eller [tuple](https://docs.python.org/3/library/stdtypes.html" \l "tuple" \o "tupel), helst i en deterministisk ordning så att provet är reproducerbart.

Här är några andra vanliga listmetoder.

* list.append(elem) -- lägger till ett enda element i slutet av listan. Vanligt fel: returnerar inte den nya listan, ändrar bara originalet.
* list.insert(index, elem) -- infogar elementet vid det givna indexet, flyttar element åt höger.
* list.extend(list2) lägger till elementen i list2 i slutet av listan. Att använda + eller += på en lista liknar att använda extend().
* list.index(elem) -- söker efter det givna elementet från början av listan och returnerar dess index. Kastar ett ValueError om elementet inte visas (använd "in" för att kontrollera utan ett ValueError).
* list.remove(elem) -- söker efter den första instansen av det givna elementet och tar bort det (kastar ValueError om det inte finns)
* list.sort() -- sorterar listan på plats (returerar den inte). (Funktionen sortered() som visas senare är att föredra.)
* list.reverse() -- vänder listan på plats (returerar den inte)
* list.pop(index) -- tar bort och returnerar elementet vid det givna indexet. Returnerar elementet längst till höger om index utelämnas (ungefär motsatsen till append()).

Lägg märke till att dessa är \*metoder\* på ett listobjekt, medan len() är en funktion som tar listan (eller strängen eller vad som helst) som ett argument.