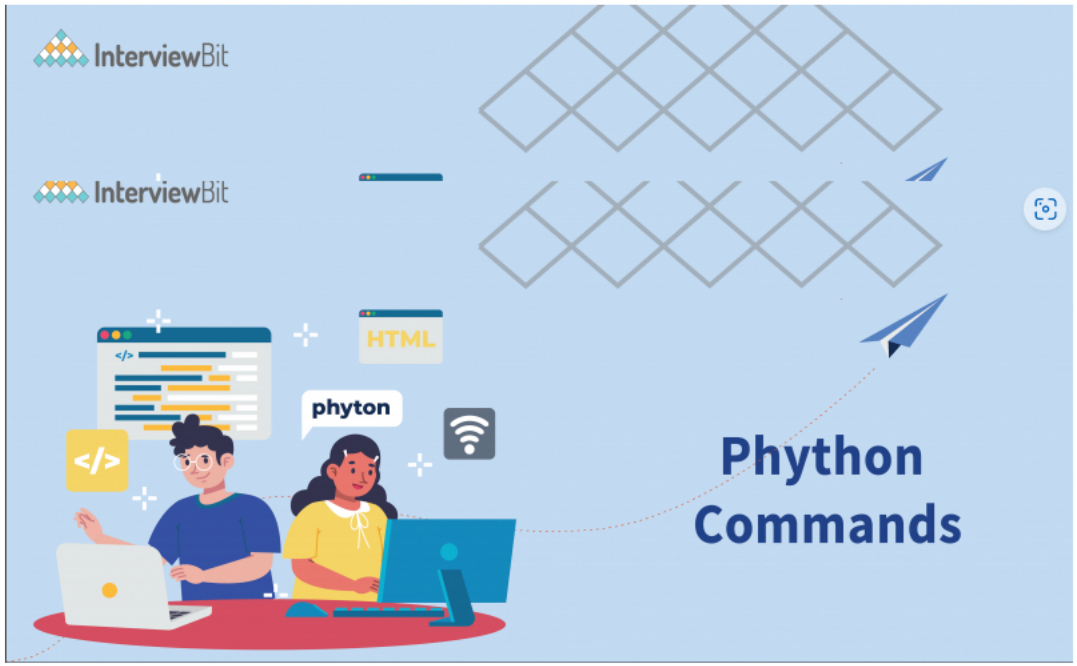
OPDRACHTEN · 11 minuten lezen

Python-opdrachtenlijst

6 juni 2022



Inhoudsopgave

Introductie

Python is een van de meest populaire programmeertalen. Het is een geïnterpreteerde algemene taal op hoog niveau. Python is ontwikkeld door Guido van Rossum in 1991. De syntaxis is zeer beginnersvriendelijk en gemakkelijk te leren. Dat is de reden waarom de meeste mensen python-taal aan beginners voorstellen als hun eerste programmeertaal om te leren. In dit bericht gaan we **de beste python-opdrachten** bespreken die uw **python-leerreis** gemakkelijker kunnen maken.

In de programmeertaal Python verwijzen opdrachten in feite naar verschillende functies of methoden die we op de python-shell kunnen uitvoeren om ze als opdrachten te laten werken. Volgens de officiële documentatie van Python zijn er geen "commando's" in Python, maar we hebben verschillende soorten functies zoals input(), type(), len(), enzovoort. Dus in dit bericht gaan we de termen commando's en functies door elkaar gebruiken. Laten we nu aan de slag gaan met de lijst met python-opdrachten en elke opdracht in detail bespreken.

Python-opdrachtenlijst

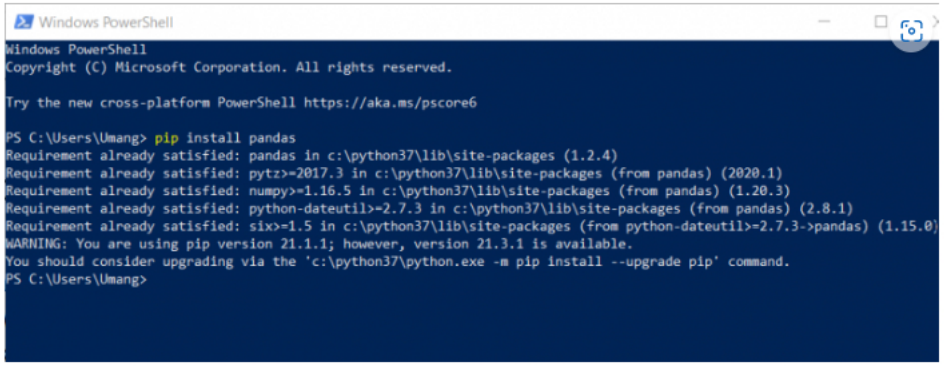
Basis Python-opdrachten die u moet kennen

Hier zijn de lijst met basis python-opdrachten.

pip install, opdracht

pip is een pakketbeheerder die is geschreven in python. Het wordt gebruikt om softwarepakketten te installeren en te beheren. De opdracht pip install wordt gebruikt om elk softwarepakket te installeren vanuit een online opslagplaats van openbare pakketten, de Python Package Index. Als u deze opdracht in Windows wilt uitvoeren, moet u uw Windows PowerShell openen en vervolgens de volgende syntaxis gebruiken om een pakket te installeren.

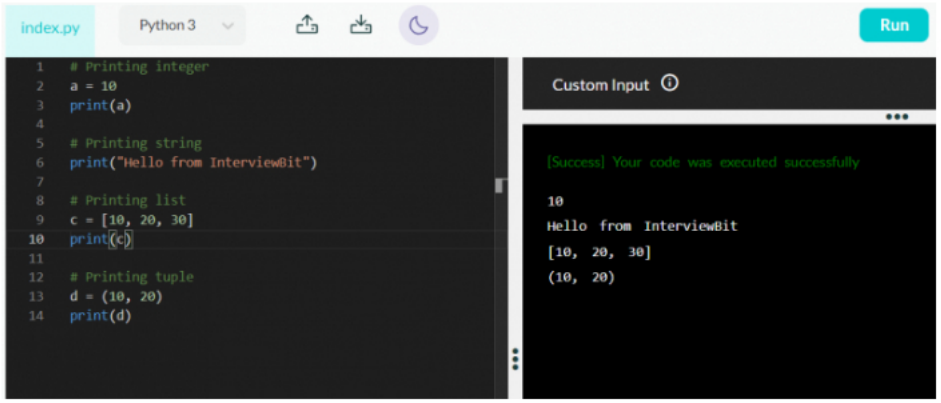
Syntaxis: pip install package-name
Voorbeeld:



afdrukken, opdracht

de opdracht afdrukken wordt gebruikt om een bericht af te drukken op het scherm of een ander standaarduitvoerapparaat. Het bericht kan een tekenreeks of een ander object zijn. De opdracht afdrukken kan worden gebruikt om elk type object af te drukken, zoals gehele getallen, tekenreeksen, lijsten, tupels, enz.

Syntaxis: print(object)
Voorbeeld:

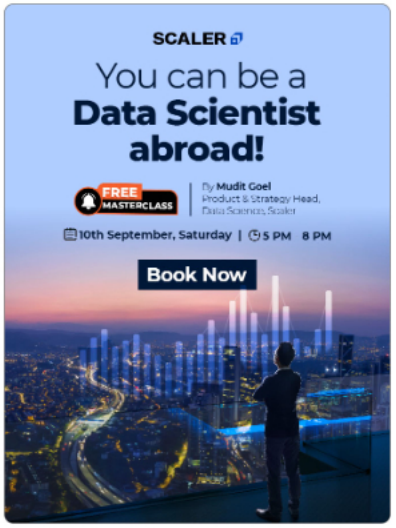


type, opdracht

de opdracht type wordt gebruikt om het type of de klasse van een object te controleren.

Syntaxis: type(object)
Voorbeeld:

```
>>> type(10)
<class 'int'>
>>> type("InterviewBit")
<class 'str'>
>>> type((10, 20, 30))
<class 'tuple'>
>>> type([10, 20, 30])
<class 'list'>
```



bereik, opdracht

de opdracht bereik wordt gebruikt om een reeks gehele getallen te genereren die standaard van 0 tot n beginnen, waarbij n niet is opgenomen in de gegenereerde getallen. We gebruiken deze opdracht meestal voor lussen.

Syntaxis: bereik (begin, stop, stap)

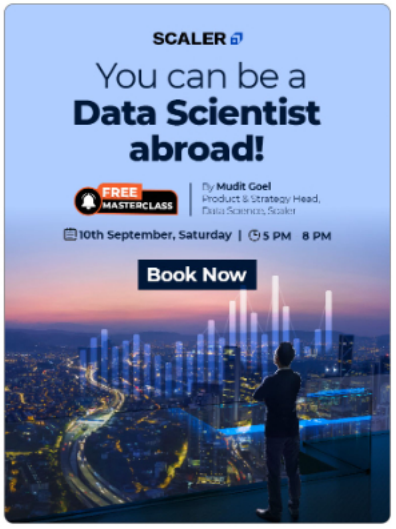
- *start* verwijst naar het begin van het bereik (optioneel; standaard 0)
- *stop* verwijst naar het nummer waarvoor u wilt stoppen (verplicht)
- *stap* verwijst naar het aantal stappen (optioneel; standaard 1)

Voorbeeld:

```
>>> for i in range(10):
        print(i)

0
1
2
3
4
5
6
7
8
9
>>>
```

```
>>> for i in range(10,20):
        print(i)

10
11
12
13
14
15
16
17
18
19
>>>
```

```
>>> for i in range(20,40,2):
        print(i)

20
22
24
26
28
30
32
34
36
38
>>>
```



Notitie: Als u twee parameters opgeeft voor de functie range(), worden deze altijd beschouwd als (start,stop) en niet als (stop,stap).

rond, opdracht

de opdracht rond wordt gebruikt om een getal af te ronden op een bepaalde precisie in decimale cijfers. Dat betekent dat als u zoveel cijfers achter de komma in een drijvende-kommanummer hebt, u de opdracht rond kunt gebruiken om het opgegeven getal af te ronden. U kunt aangeven hoeveel cijfers u achter de komma wilt hebben.

Syntaxis: afgerond(getal, cijfers)

- *getal* verwijst naar het drijvende-kommagetal.
- *cijfers* verwijzen naar het aantal cijfers dat u achter de komma wilt hebben. (Optioneel; Standaard 0)

Voorbeeld:

```
Python 3.7.1 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 
4)] on win32
Type "help", "copyright", "credits" o
>>> round(23.5678,2)
23.57
>>> round(23.5624,2)
23.56
>>> round(23.78)
24
>>> round(23.45)
23
>>> round(23.55)
24
>>>
```


invoer, opdracht

de opdracht input wordt gebruikt om input van de gebruiker te ontvangen. De stroom van het programma wordt gestopt totdat de gebruiker een waarde heeft ingevoerd. Wat de gebruiker ook invoert, het wordt door de invoerfunctie omgezet in een tekenreeks. Als u een geheel getal als invoer wilt nemen, moet u het expliciet converteren.

Syntaxis: invoer(bericht)

bericht verwijst naar de tekst die u aan de gebruiker wilt weergeven. (Optioneel)

Voorbeeld:

```
>>> input("Enter value: ")
Enter value: 23
'23'
>>> input("Enter value: ")
Enter value: Hello
'Hello'
>>> var = input("Enter value: ")
Enter value: 45
>>> type(var)
<class 'str'>
>>>
```

len, opdracht

de opdracht len of de functie len() wordt gebruikt om het aantal items in een object op te halen. Als het object een tekenreeks is, retourneert de functie len() het aantal tekens dat erin aanwezig is. Als het object een lijst of tupel is, retourneert het het aantal elementen dat in die lijst of tupel aanwezig is. len() geeft een foutmelding als u probeert er een geheel getal aan door te geven.

Syntaxis: len(object)

object waarvan u de lengte wilt vinden (verplicht)

Voorbeeld:

```
>>> var = "Hello World"
>>> len(var)
11
>>> var2 = [10,20,30,40]
>>> len(var2)
4
>>> var3 = (10,20,30,40)
>>> len(var3)
4
>>> var4 = 4567
>>> len(var4)
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    len(var4)
TypeError: object of type 'int' has no len()
>>>
```

Loop-opdrachten

In python hebben we twee primitieve luscommando's namelijk while en for. De opdracht *while* loop wordt gebruikt om een reeks instructies uit te voeren zolang de gegeven voorwaarde waar is.

Syntaxis van while loop: while condition:
instructies
update iterator

Voorbeeld:

```
>>> x = 1
>>> while x < 10:
        print(x)
        x = x + 1

1
2
3
4
5
6
7
8
9
>>>
```

De opdracht for loop wordt gebruikt om een set instructies uit te voeren door over een reeks te herhalen. Deze volgorde kan een lijst, tupel, string, woordenboek, etc. zijn.

Syntaxis van voor lus: voor x in volgorde:
instructies

Voorbeeld:

```
>>> names = ["A", "B", "C", "D"]
>>> for x in names:
        print(x)


A
B
```

```
C
D
>>>
```

Tussenliggende Python-opdrachten

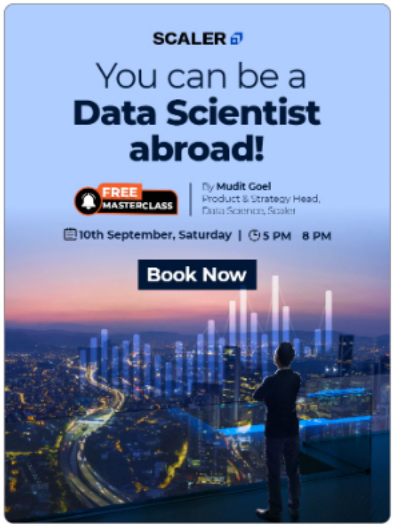
Tekenreeksopdrachten

In de programmeertaal python hebben we verschillende tekenreeksopdrachten die we kunnen gebruiken op tekenreeksobjecten. Deze opdrachten wijzigen het oorspronkelijke tekenreeksobject niet, ze retourneren alleen een nieuw object. Enkele van de belangrijkste stringfuncties zijn:

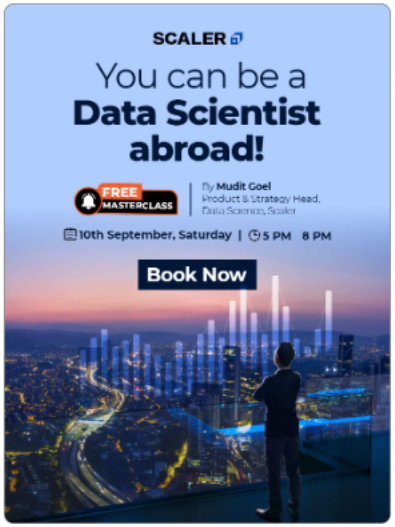
isalnum(): Hiermee wordt gecontroleerd of alle tekens van een bepaalde tekenreeks alfanumeriek zijn of niet. Het retourneert een Booleaanse waarde.

Syntaxis: stringnaam.isalnum()

Voorbeeld:




```
>>> str = "123Hello"
>>> str.isalnum()
True
>>> str = "123@#Hello"
>>> str.isalnum()
False
```



kapitaliseren()

De functie capitalize() wijzigt het eerste teken van de tekenreeks in hoofdletters als dit kleine letters zijn. Als het eerste teken hoofdletters of een geheel getal of een speciaal teken is, doet het niets.

Syntaxis: stringnaam.capitalize()

Voorbeeld:

```
>>> str = "hello"
>>> str.capitalize()
```

```
'Hello'
>>> str = "Hello"
>>> str.capitalize()
'Hello'
>>> str = "123"
>>> str.capitalize()
'123'
>>>
```

vinden()

de opdracht find() wordt gebruikt om te zoeken naar een subtekenreeks in een tekenreeks. Het retourneert de index van het eerste optreden van de subtekenreeks als deze aanwezig is, anders retourneert het -1.

Syntaxis: string.find(subtekenreeks)

- *tekenreeks* verwijst naar de tekenreeks waarin u wilt zoeken.
- *subtekenreeks* verwijst naar de waarde die u wilt doorzoeken.

Voorbeeld:

1. count(): count() functie retourneert het aantal voorkomens van een subtekenreeks in een stringobject.

Syntaxis: stringnaam.count(subtekenreeks, begin, einde)

- *stringname* verwijst naar de tekenreeks waarin u wilt zoeken.
- *subtekenreeks* verwijst naar de waarde waarvan u het aantal wilt vinden.
- *start* verwijst naar die beginindex binnen de tekenreeks waar de zoekopdracht begint (optioneel)
- *einde* verwijst naar die eindindex binnen de tekenreeks waar de zoekopdracht eindigt (optioneel)

Voorbeeld:

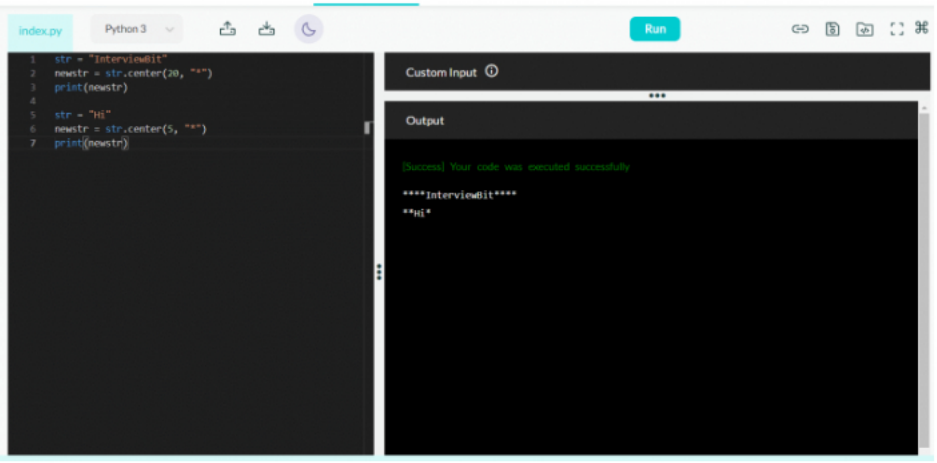
center()

de opdracht center wordt gebruikt om een tekenreeks in het midden uit te lijnen, waarbij een opgegeven teken (standaard is dit spatie) als vulteken wordt gebruikt.

Syntaxis: string.center(lengte, teken)

- *tekenreeks* is de tekenreeks die u in het midden wilt uitlijnen
- *lengte* is de volledige lengte die wordt genomen door de nieuwe snaar waarin beide zijden per *teken* worden gevuld en in het midden hebben we de originele snaar.
- *teken* verwijst naar het teken dat wordt gebruikt om de ontbrekende ruimte aan elke kant te vullen. Standaard is " " (spatie).

Voorbeeld:



Lijstopdrachten

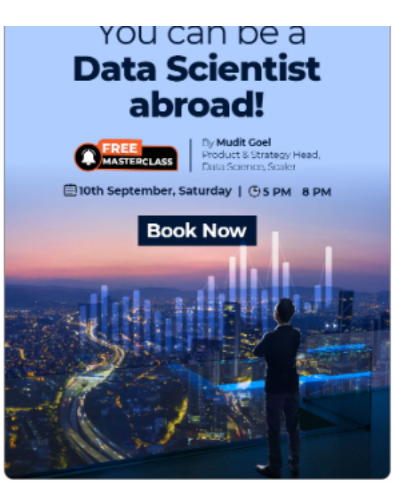
Net als een string hebben we ook verschillende commando's voor lijstobjecten. Lijsten worden gebruikt voor meerdere elementen in één object. We kunnen lijsten gebruiken om elementen van verschillende gegevenstypen op te slaan. Enkele van de belangrijkste lijstmethoden zijn:

append(): de opdracht append wordt gebruikt om een element aan het einde van de lijst toe te voegen.

Syntaxis: list.append(element)

- *list* is het lijstobject waaraan u een element wilt toevoegen
- *element* verwijst naar het nieuwe item dat u aan de lijst wilt toevoegen

Voorbeeld:

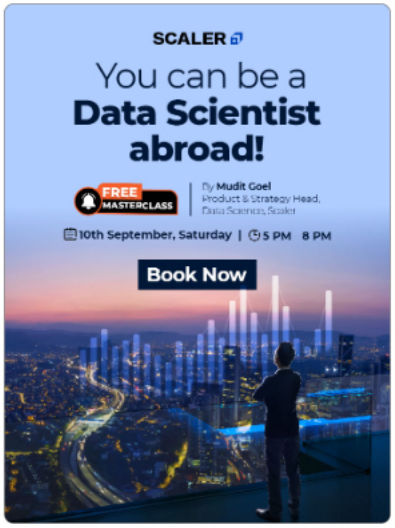


kopie()

copy() wordt gebruikt om een nieuwe kopie van het list-object te maken. Het retourneert een nieuw lijstobject.

Syntaxis: list.copy()

Voorbeeld:



invoegen()

de opdracht invoegen wordt gebruikt om een element toe te voegen op een opgegeven positie in het lijstobject. Het neemt twee parameters positie en element.

Syntaxis: listname.insert(positie, element)

- *positie* waarop u een nieuw element wilt invoegen. Als u een positie opgeeft die groter is dan het aantal elementen in de lijst, wordt deze altijd aan het einde ingevoegd.
- *element* verwijst naar het nieuwe element dat moet worden toegevoegd.

Voorbeeld:

pop()

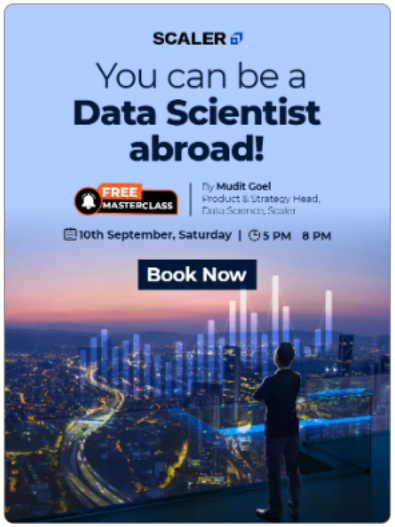
pop()

de methode pop() wordt gebruikt om een element van een opgegeven positie in de lijst te verwijderen. Het retourneert het element nadat het uit de lijst is verwijderd.

Syntaxis: listname.pop(positie)

positie waaruit u het element wilt verwijderen.

Voorbeeld:

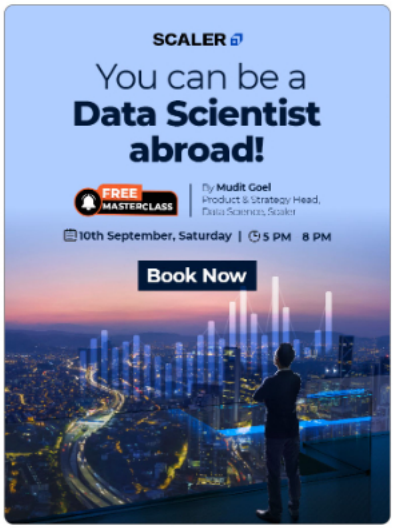


omgekeerd()

reverse-methode keert de volgorde van alle elementen in de lijst om. Het wijzigt het oorspronkelijke lijstobject en retourneert niets.

Syntaxis: list.reverse()

Voorbeeld:



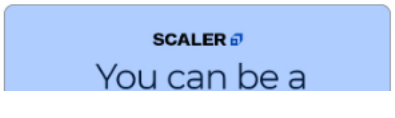
sorteren()

de sorteermethode wordt standaard gebruikt om de elementen van de lijst in oplopende volgorde te sorteren.

Syntaxis: list.sort()

Voorbeeld:

Tuple-opdrachten

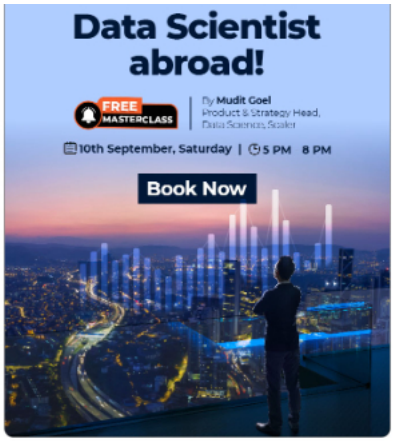


Tuple is een ingebouwd gegevenstype dat wordt gebruikt om meerdere elementen in één variabele op te slaan. Tuple-objecten zijn geordend en onveranderlijk. Er zijn slechts twee ingebouwde tupelmethoden die als volgt zijn:

count(): telmethode wordt gebruikt om de voorkomens van een element in de tupel te tellen.

Syntaxis: tupel.count(element)

Voorbeeld:



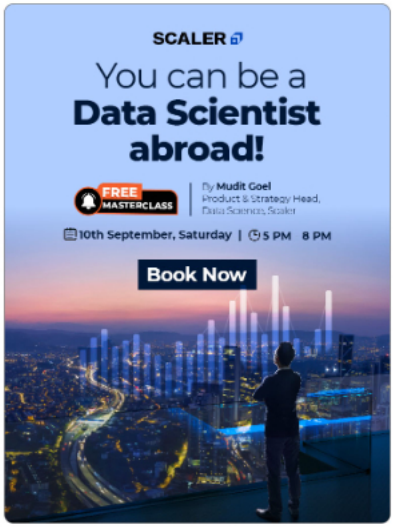
index()

Deze methode wordt gebruikt om de index van het eerste voorkomen van een element te vinden. Als het element niet in de hele tupel wordt gevonden, zal het een ValueError verhogen.

Syntaxis: tuple.index(element)

- *tuple* is het tuple-object waarin u een element wilt zoeken
- *-element* verwijst naar het item dat u wilt doorzoeken

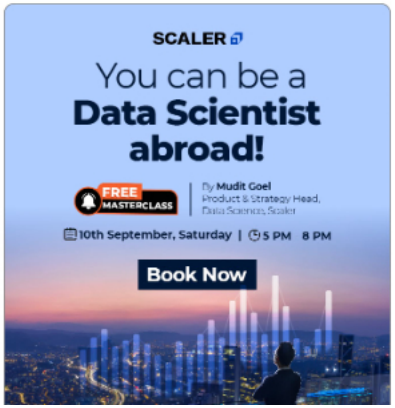
Voorbeeld:



Lijst met geavanceerde Python-opdrachten

Opdrachten instellen

Sets worden ook gebruikt om meerdere elementen in één object op te slaan, maar ze staan geen dubbele elementen toe. Sets zijn ongeordend en niet-geïndexeerd. Dat betekent dat als u alle elementen van een set afdrukt, ze in willekeurige volgorde worden afgedrukt. Zodra de set is gemaakt, kunt u de elementen niet



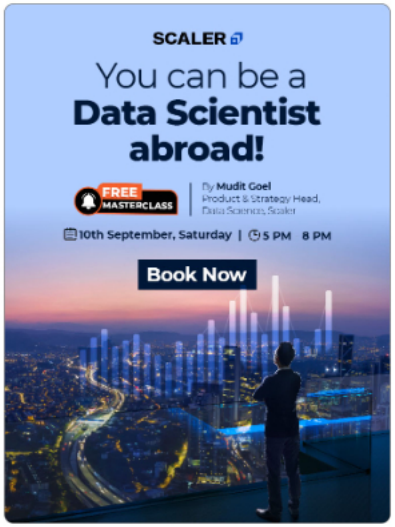
wijzigen, maar u kunt nieuwe elementen toevoegen of ook bestaande elementen verwijderen. Laten we nu enkele belangrijke setopdrachten bespreken die python biedt als ingebouwde methoden.

add(): de opdracht add wordt gebruikt om een nieuw element aan de set toe te voegen.

Syntaxis: setnaam.toevoegen(element)

- *setname* is de naam van de setvariabele waaraan u een nieuw element wilt toevoegen.
- *-element* verwijst naar het nieuwe item dat u wilt toevoegen.

Voorbeeld:

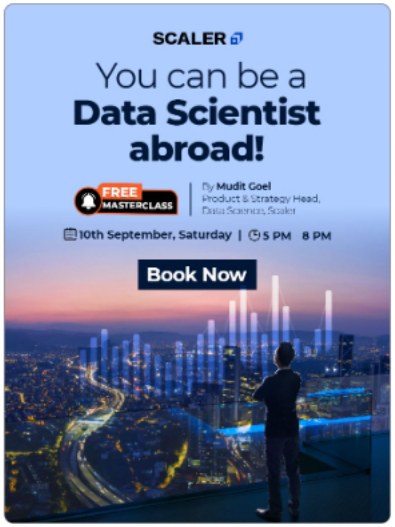


clear()

clear verwijdert alle elementen van een set. Er zijn geen parameters voor nodig.

Syntaxis: setnaam.clear()

Voorbeeld:

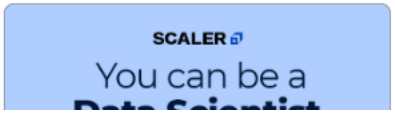


teruggooien()

discard wordt gebruikt om het opgegeven element uit de set te verwijderen. Als het opgegeven element niet in de set wordt gevonden, geeft het geen fout.

Syntaxis: setnaam.discard(element)

- *setname* is de naam van de setvariabele waaruit u een element wilt verwijderen.
- *element* verwijst naar het element dat u wilt verwijderen.



Voorbeeld:

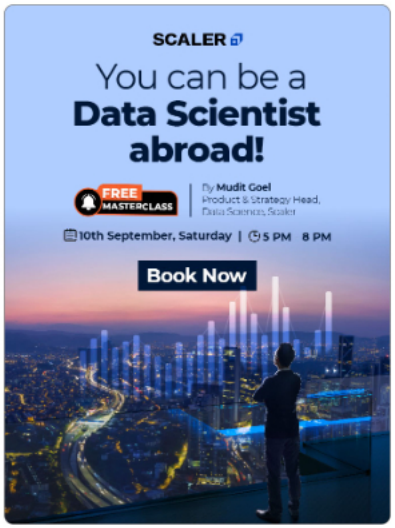
remove()

De opdracht remove wordt ook gebruikt om een opgegeven element uit de set te verwijderen, maar het verschilt van verwijderen omdat remove een fout geeft als het opgegeven element niet in de set wordt gevonden.

Syntaxis: setnaam.remove(element)

- *setname* is de naam van de setvariabele waaruit u een element wilt verwijderen.
- *element* verwijst naar het element dat u wilt verwijderen.

Voorbeeld:



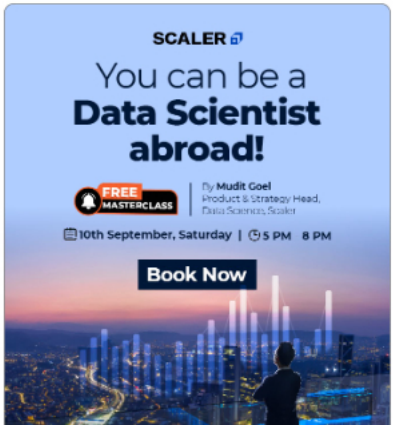
verschil()

verschilmethode wordt gebruikt om een set te krijgen die het verschil van twee sets bevat. Het verschil in verzamelingen betekent dat het alleen die elementen heeft die slechts in de ene verzameling aanwezig zijn en niet in een andere verzameling. Stel dat we twee sets A en B hebben. Set A heeft {1,2,3} en set B heeft {2, 4, 6}. Dan is het verschil tussen A en B {1,3}.

Syntaxis: setA.verschil(setB)

- Elementen van setB worden verwijderd uit setA indien aanwezig.

Voorbeeld:



difference_update()

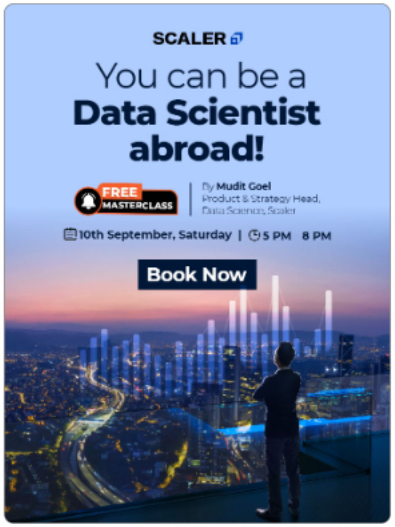
difference_update()

difference_update methode wordt gebruikt om een set elementen te krijgen die aanwezig zijn in de eerste set en niet gebruikelijk zijn in beide sets. Dat betekent dat het de elementen verwijdert die in beide sets bestaan. Het retourneert geen nieuwe set, het verwijdert alleen gemeenschappelijke elementen uit de eerste set.

Syntaxis: setA.difference_update(setB)

- Elementen die zowel in setA als in setB voorkomen, worden uit setA verwijderd.

Voorbeeld:



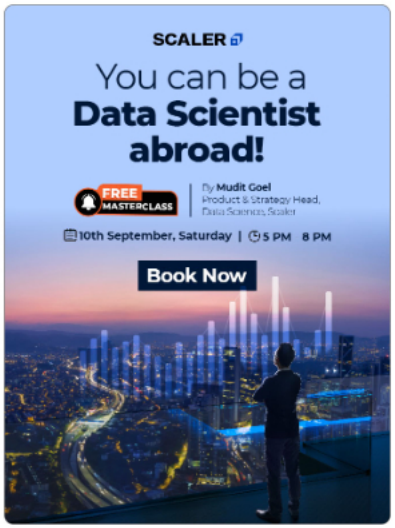
kruispunt()

de snijpuntmethode retourneert een verzameling met elementen die in alle opgegeven sets voorkomen.

Syntaxis: set.intersection(set1, set2, ... setn)

- Set van elementen die bestaan in de set, set1, set2, ... setn wordt geretourneerd.

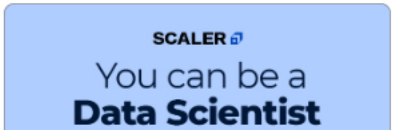
Voorbeeld:



issubset()

issubsetmethode controleert of alle elementen van verzameling A al dan niet aanwezig zijn in verzameling B. Het retourneert een Booleaanse waarde.

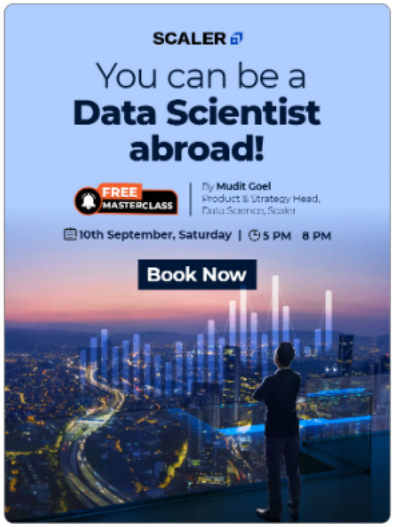
Syntaxis: setA.issubset(setB)



Voorbeeld:

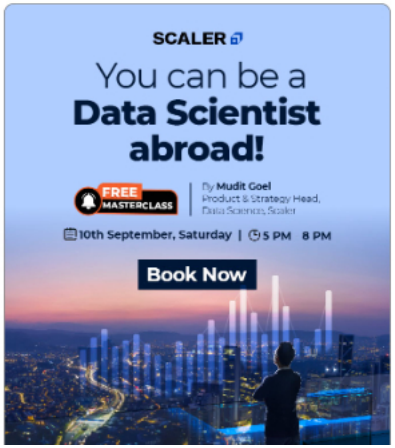
symmetric_difference()

Deze methode retourneert een verzameling die elementen van beide verzamelingen bevat, behalve de verzamelingen die in beide sets gebruikelijk zijn.



Syntaxis: setA.symmetric_difference(setB)

Voorbeeld:



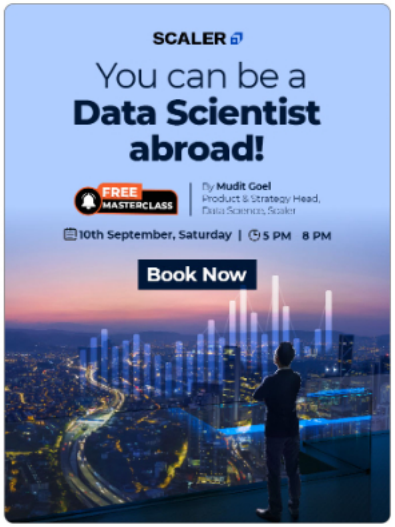
unie()

union-methode retourneert een verzameling die alle elementen van beide

verzamelingen bevat, behalve de gedupliceerde elementen.

Syntaxis: setA.union(setB)

Voorbeeld:



Woordenboekopdrachten

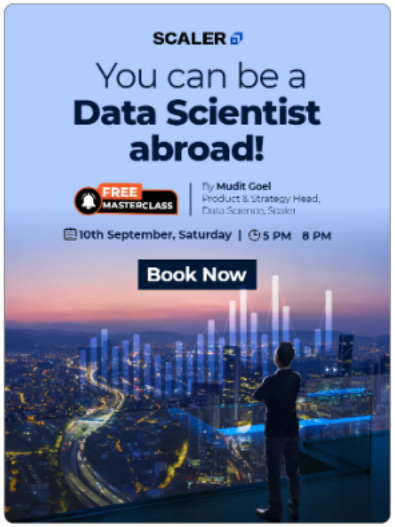
Woordenboek is een ingebouwd type in Python. Het wordt gebruikt om sleutel-waardeparen op te slaan. Het is geordend, wijzigbaar en staat geen dubbele sleutels toe. Dat betekent dat we in een woordenboek geen twee paren kunnen toevoegen met dezelfde waarde van sleutels. Python biedt een reeks ingebouwde methoden die we kunnen gebruiken op woordenboekobjecten.

fromkeys(): methode fromkeys() wordt gebruikt om een woordenboek te genereren met opgegeven sleutels en een opgegeven waarde.

Syntaxis: dict.fromkeys(sleutels, waarde)

- *toetsen* is de tupel of lijst van sleutelelementen.
- *waarde* verwijst naar de waarde die zou worden gekoppeld aan alle opgegeven sleutels.

Voorbeeld:

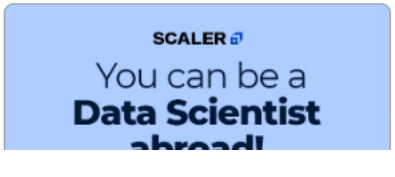


krijgen()

get-methode wordt gebruikt om een waarde van de opgegeven sleutel te krijgen. Als een sleutel niet in het woordenboek wordt gevonden, retourneert deze niets, tenzij we iets opgeven in de parameters.

Syntaxis: dictionary.get(sleutel, waarde)

- *woordenboek* is de naam van het woordenboekobject waarin u wilt zoeken.



- *sleutel* verwijst naar de sleutel die u in het woordenboek wilt zoeken.
- waarde is de waarde die zou worden geretourneerd als de sleutel niet in het woordenboek wordt gevonden.

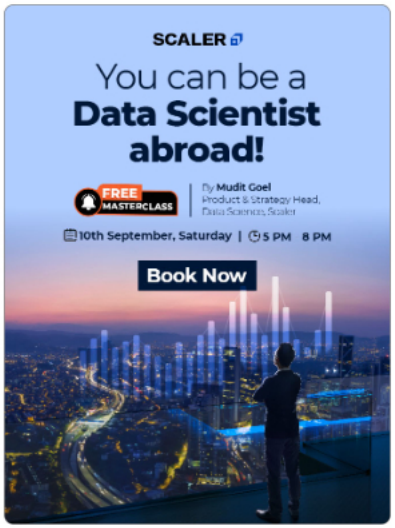
Voorbeeld:

artikelen()

de methode items wordt gebruikt om de woordenboekelementen weer te geven. Het geeft alle sleutel-waardeparen weer die in het woordenboek aanwezig zijn. Het retourneert een weergaveobject dat alle sleutel-waardeparen als tupels in een lijst bevat. Er zijn geen parameters voor nodig.

Syntaxis: dictionary.items()

Voorbeeld:

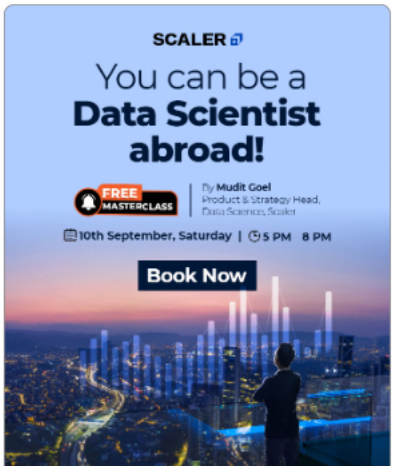


sleutels()

keys methode wordt gebruikt om alle sleutels aanwezig in het woordenboek te krijgen. Het retourneert een weergaveobject dat alle sleutels van de woordenlijst bevat als een lijst. Er zijn geen parameters voor nodig.

Syntaxis: dictionary.keys()

Voorbeeld:



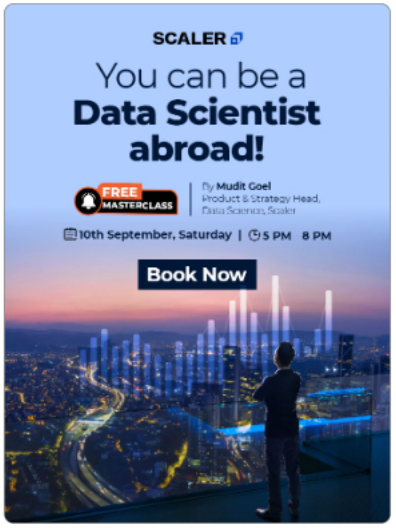
waarden()

de methode waarden wordt gebruikt om alle waarden in het woordenboek op te

halen. Het retourneert een weergaveobject dat alle waarden van de woordenlijst bevat als een lijst. Er zijn geen parameters voor nodig.

Syntaxis: dictionary.values()

Voorbeeld:



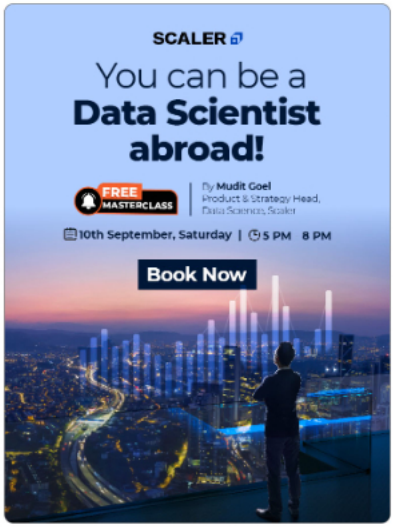
pop()

pop-methode wordt gebruikt om een sleutel-waardepaar uit het woordenboek te verwijderen door de sleutel op te geven. Het retourneert de waarde van het sleutel-waardepaar dat moet worden verwijderd.

Syntaxis: dictionary.pop(sleutel)

- *sleutel* verwijst naar de sleutel van het paar dat u uit het woordenboek wilt verwijderen.

Voorbeeld:

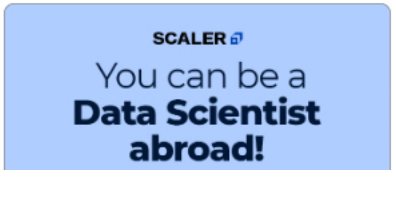


popitem()

de opdracht popitem wordt gebruikt om het laatst ingevoegde paar uit het woordenboek te verwijderen. Er zijn geen parameters voor nodig. Het retourneert het verwijderde paar als een tupel.

Syntaxis: dictionary.popitem()

Voorbeeld:



setdefault()

de methode setdefault wordt gebruikt om de waarde van een opgegeven sleutel op te halen. Als de sleutel niet bestaat, wordt de sleutel ingevoegd met de waarde die als parameter is doorgegeven. Als u geen waarde opgeeft, wordt de sleutel met de waarde Geen ingevoegd.

Syntaxis: dictionary.setdefault(sleutel, waarde)

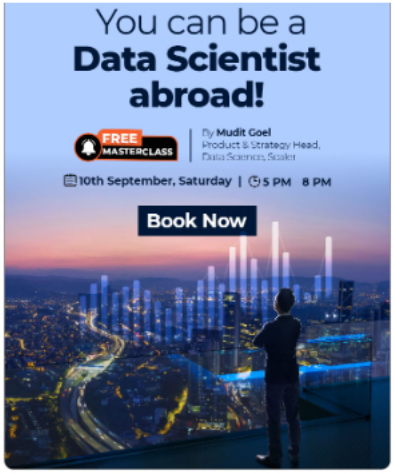
Voorbeeld:

Conclusie

In dit bericht hebben we de beste python-opdrachten besproken die elke Python-programmeur zou moeten leren. Je moet zeker elk commando zelf proberen. Probeer ook te experimenteren met willekeurige invoerparameters om het gedrag van opdrachten te zien. Python-opdrachten zijn gemakkelijk te gebruiken, gemakkelijk te schrijven en gemakkelijk te leren. U hoeft niet alle opdrachten te onthouden, maar u moet bekend zijn met wat alle opdrachten als functionaliteit bieden.

Veelgestelde vragen

V. Wat is de opdracht Python 3?
A. Python 3-opdrachten worden uitgebracht in 2008. Deze opdrachten worden geïntroduceerd met Python 2-functies en ze zijn ook compatibel met Python 2. Python 3-opdrachten zijn intuïtiever voor programmeurs en nauwkeuriger terwijl ze het resultaat leveren.



V. Hoe gebruik je Python-commando's?
Een. Om Python-opdrachten te gebruiken of uit te voeren, kunt u python-shell gebruiken. Python Shell is de python-interpreter die wordt gebruikt om eenvoudige Python-programma's uit te voeren en python-opdrachten uit te voeren. Het biedt gewoon een snelle manier om opdrachten uit te voeren zonder een bestand te maken.

V. Wat zijn enkele veelvoorkomende Python-opdrachten?

Een. Enkele veel voorkomende Python-opdrachten zijn invoer,afdrukken, bereik, rond, pip install, len, sort, loop commando's zoals voor en terwijl ga zo maar door enzovoort.

V. Wat zijn magische commando's in Python?
A. Magic-opdrachten zijn snelkoppelingen of verbeteringen ten opzichte van de gebruikelijke Python-syntaxis. Deze opdrachten zijn ontworpen om routinetaken te vergemakkelijken. Ze stellen ons in staat om eenvoudig het gedrag van het IPython-systeem te controleren en verschillende veelvoorkomende problemen in standaardgegevensanalyse op te lossen, bijvoorbeeld het uitvoeren van een extern script of het berekenen van de uitvoeringstijd van een stuk code.

Aanvullende bronnen

InterviewBit Blog
Praktijk codering
Python MCQ
Python Interview Vragen
PHP vs Python
Python Ontwikkelaar Hervatten
Python Applicaties
Python Developer Skills

Python | Python-opdrachten

VORIGE POST

VAARDIGHEDEN
Top android developer vaardigheden die u moet weten
maart 4, 2022

VOLGEND BERICHT

OPDRACHTEN
Top Docker-opdrachten met voorbeelden
februari 8, 2022

Categorieën

OPDRACHTEN · 7 minuten lezen

Top Docker-opdrachten met voorbeelden

februari 8, 2022

Introductie

Docker is een open source linux gebaseerd containerisatieplatform waarmee ontwikkelaars hun applicatie in containers kunnen verpakken, Containers zijn de uitvoerbare componenten die de broncode van de toepassing combineren met de

bibliotheken van het besturingssysteem en afhankelijkheden die nodig zijn om de code in elke omgeving uit te voeren. Het voordeel is dat ontwikkelaars zich geen zorgen hoeven te maken of hun code op dezelfde manier wordt uitgevoerd als op hun machine. Docker wint in de loop der jaren aan populariteit en is nu een must have skill. Deze blog leert u over de docker-opdrachten die vaak worden gebruikt. Voordat u naar de docker-opdrachten en de docker-opdrachtenlijst gaat, is het essentieel om een goed begrip te hebben van waar docker over gaat.

Wat is Docker?

Bij het werken aan een groot project werken verschillende ontwikkelaars samen. Er worden meerdere teams gevormd om een project op te bouwen. Op grote schaal kunnen we bedenken dat er een ontwikkelteam is dat de applicaties gaat ontwikkelen, een testteam dat de applicaties test en een operations team dat de applicatie op de productieserver implementeert. Het proces verschilt van bedrijf tot bedrijf.

Als ontwikkelaar gebruik je misschien een aantal frameworks, zoals Spring framework of Django framework. Dit framework heeft een aantal afhankelijkheden nodig, de afhankelijkheden variëren op hun beurt van besturingssystemen tot besturingssystemen en van versie tot versie. Dat betekent dat een code die met succes is gecompileerd in versie 1.0.1 mogelijk niet hetzelfde wordt uitgevoerd in versie 1.0.2. Dit is een problematische kwestie. De eerste oplossing is om elke afhankelijkheid en specificaties in een zip-bestand of een jar-bestand te geven terwijl het naar een ander team wordt verzonden. Het probleem hier is dat sommige afhankelijkheden afhankelijk zijn van het besturingssysteem. U kunt niet uw volledige besturingssysteem aan een andere persoon geven.

De oplossing is het gebruik van een Hypervisor. Een hypervisor, ook wel een virtuele-machinemonitor of VMM genoemd, is software waarmee virtuele machines (VM's) worden gemaakt en uitgevoerd. Met een hypervisor kan één hostcomputer meerdere gast-VM's ondersteunen door de bronnen virtueel te delen tussen VM's, zoals geheugen en verwerking. Denk aan een Hypervisor als volgt: U hebt uw computerhardware en daarbovenop is er een besturingssysteem, u kunt een

Hypervisor in uw systeem installeren en er een gastbesturingssysteem op uitvoeren. Het beste deel is dat u uw besturingssysteem eenvoudig kunt verzenden door de virtuele afbeelding naar een andere persoon te maken. Zodra de andere persoon een afbeelding heeft ontvangen, kan deze de instantie maken en uitvoeren. Dit is het concept van virtualisatie.

Het maken van een nieuwe virtuele machine elke keer voor een nieuwe toepassing wordt echter omslachtig. Vanaf hier komt het concept van containerisatie in beeld. Containerisatie is de evolutie van virtualisatie. Terwijl virtualisatie zich richt op de distributie van besturingssystemen, richt containerisatie zich aan de andere kant op het opsplitsen van besturingssystemen in brokken die effectiever kunnen worden gebruikt.

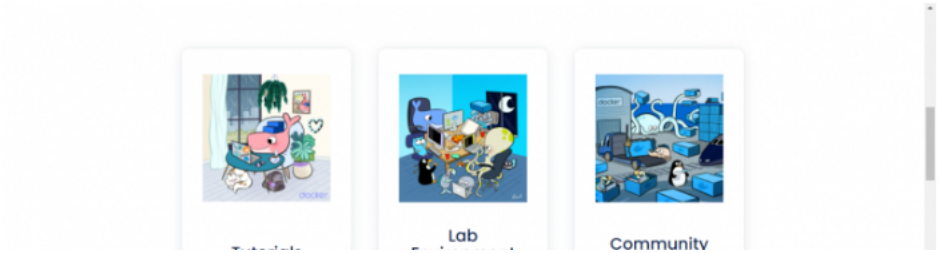
In een notendop **zijn containers een oplossing voor het probleem van hoe software betrouwbaar kan worden uitgevoerd wanneer deze van de ene computeromgeving naar de andere wordt verplaatst.**

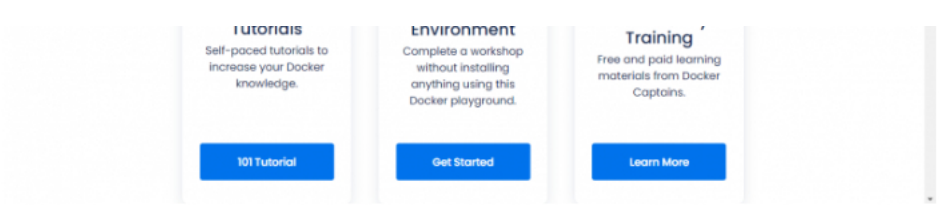
Top Docker-opdrachten

Voordat u naar de lijst met docker-opdrachten gaat, is het essentieel dat u docker in uw systemen hebt geïnstalleerd. Als u het al hebt geïnstalleerd, kunt u doorgaan. Als u docker niet hebt geïnstalleerd, kunt u dit eenvoudig doen door de stappen in de documentatie te volgen.

Als u net begint en geen docker wilt instellen, kunt u de online docker-speeltuin gebruiken, Spelen met Docker, Het is een project dat wordt gesponsord door docker om het voor nieuwe studenten gemakkelijk te maken om aan de slag te gaan met docker. Er is een laboratoriumomgeving waar u een online terminal krijgt om docker-opdrachten gratis uit te voeren. Het is eenvoudig en gemakkelijk te gebruiken.

Laten we de verschillende docker-opdrachten bekijken





docker –versie

Deze opdracht geeft standaard de versie-informatie weer in een gemakkelijk te lezen lay-out, U kunt ook uw eigen indeling opgeven.

```
$
$
$
$
$
$ docker --version
Docker version 18.03.1-ce, build 9ee9f40
$
```

De opdracht kan op verschillende manieren worden gebruikt. Enkele van de manieren worden hieronder gegeven

- Als u de serverversie wilt ophalen, gebruikt u de volgende opdracht

docker versie –formaat '{{. Server.Version}}'

```
$
$ docker version --format '{{.Server.Version}}'
18.03.0-ce
$
```

- Als u de onbewerkte json-gegevens wilt dumpen, gebruikt u de volgende opdracht

docker versie –format '{{json .}}'

```
$ docker version --format '{{json .}}'
{"Client":{"Platform":{"Name":""},"Version":"18.03.1-ce","ApiVersion":"1.
37","DefaultAPIVersion":"1.37","GitCommit":"9ee9f40","GoVersion":"go1.9.2
```

```
","Os":"linux","Arch":"amd64","BuildTime":"Thu Apr 26 07:12:25 2018","Exp
erimental":false,"Orchestrator":"swarm"},"Server":{"Platform":{"Name":""}
,"Components":[{"Name":"Engine","Version":"18.03.0-ce","Details":{"ApiVer
sion":"1.37","Arch":"amd64","BuildTime":"Fri Mar 23 01:48:12 2018","Exper
imental":"false","GitCommit":"0520e24302","GoVersion":"go1.10","KernelVer
sion":"4.14.29-1-lts","MinAPIVersion":"1.12","Os":"linux"}}],"Version":"1
8.03.0-ce","ApiVersion":"1.37","MinAPIVersion":"1.12","GitCommit":"0520e2
4302","GoVersion":"go1.10","Os":"linux","Arch":"amd64","KernelVersion":"4
.14.29-1-lts","BuildTime":"2018-03-23T01:48:12.000000000+00:00"}}
$
```


docker zoeken

Deze opdracht wordt gebruikt voor het zoeken in de docker-hub naar openbare afbeeldingen. De geretourneerde informatie bevat afbeeldingsnaam, beschrijving, sterren, geautomatiseerd en officieel, samen met vele andere details.

docker zoeken <imageName>

De opdracht docker search MySQL is een voorbeeld waarbij we zoeken naar MySQL-afbeeldingen, de opdracht retourneert de beschikbare afbeeldingen van MySQL op docker hub zoals hieronder weergegeven.


```
ersion":"1.37","Arch":"amd64","BuildTime":"Fri Mar 23 01:48:12 2018","Experimental":"false","GitCommit":"0520e24302","GoV
ersion":"go1.10","KernelVersion":"4.14.29-1-lts","MinAPIVersion":"1.12","Os":"linux"}}],"Version":"18.03.0-ce","ApiVersio
n":"1.37","MinAPIVersion":"1.12","GitCommit":"0520e24302","GoVersion":"go1.10","Os":"linux","Arch":"amd64","KernelVersion
":"4.14.29-1-lts","BuildTime":"2018-03-23T01:48:12.000000000+00:00"}}
$ docker search MySQL
NAME                              DESCRIPTION                                     STARS               OFFICIAL
 AUTOMATED
mysql                             MySQL is a widely used, open-source relation…   12002               [OK]

mariadb                           MariaDB Server is a high performing open sou…   4597                [OK]

mysql/mysql-server                Optimized MySQL Server Docker images. Create…   899
 [OK]
percona                           Percona Server is a fork of the MySQL relati…   568                 [OK]

phpmyadmin                        phpMyAdmin - A web interface for MySQL and M…   432                 [OK]

mysql/mysql-cluster               Experimental MySQL Cluster Docker images. Cr…   92

centos/mysql-57-centos7           MySQL 5.7 SQL database server                   92

centurylink/mysql                 Image containing mysql. Optimized to be link…   59
 [OK]
databack/mysql-backup             Back up mysql databases to... anywhere!         54

prom/mysqld-exporter                                                              46
 [OK]
deitch/mysql-backup               REPLACED! Please use http://hub.docker.com/r…   40
 [OK]
linuxserver/mysql                 A Mysql container, brought to you by LinuxSe…   35
```


U kunt de opdracht wijzigen om een niet-afgekapte beschrijving weer te geven met –no trunc zoals hieronder weergegeven:

docker zoeken –filter=stars=3 –no-trunc MySQL

De opdracht geeft bijvoorbeeld de afbeeldingen van MySQL weer met ten minste 3 sterren met een niet-afgekapte beschrijving.

```
$
$
$
$
$ docker search --filter=stars=3 --no-trunc MySQL
NAME                                      DESCRIPTION                                     STARS               OFFICIAL            AUTOMATED
mysql                                     MySQL is a widely used, open-source relational database management system (RDBMS).   12002               [OK]
mariadb                                   MariaDB Server is a high performing open source relational database, forked from MySQL.   4597                [OK]
mysql/mysql-server                        Optimized MySQL Server Docker images. Created, maintained and supported by the MySQL te   899
am at Oracle                                                                              [OK]
percona                                   Percona Server is a fork of the MySQL relational database management system created by   568
Percona.                                                                                  [OK]
phpmyadmin                                phpMyAdmin - A web interface for MySQL and MariaDB.   432                 [OK]
centos/mysql-57-centos7                   MySQL 5.7 SQL database server   92
mysql/mysql-cluster                       Experimental MySQL Cluster Docker images. Created by the MySQL team at Oracle   92
centurylink/mysql                         Image containing mysql. Optimized to be linked to another image/container.   59                  [OK]
databack/mysql-backup                     Back up mysql databases to... anywhere!   54
prom/mysqld-exporter                                                                      46                  [OK]
deitch/mysql-backup                       REPLACED! Please use http://hub.docker.com/r/databack/mysql-backup instead.   40                  [OK]
tutum/mysql                               Base docker image to run a MySQL database server
```


Wanneer u de zoekopdracht gebruikt, worden standaard 25 afbeeldingen geretourneerd. U kunt het resultaat dat wordt geretourneerd door een zoekopdracht beperken met de vlag **-limit**. De waarden kunnen tussen 1 en 100 liggen.

Om de afbeeldingen van MySQL op basis van geautomatiseerde builds uit te filteren, gebruikt u het volgende commando.

docker zoeken –filter is-automated=true MySQL

```
$ docker search --filter is-automated=true MySQL
NAME                                      DESCRIPTION                                     STARS               OFFICIAL            AUTOMATED
mysql/mysql-server                        Optimized MySQL Server Docker images. Create…   899                 [OK]
centurylink/mysql                         Image containing mysql. Optimized to be link…   59                  [OK]
prom/mysqld-exporter                                                                      46                  [OK]
deitch/mysql-backup                       REPLACED! Please use http://hub.docker.com/r…   40                  [OK]
schickling/mysql-backup-s3                Backup MySQL to S3 (supports periodic backup…   31                  [OK]
arey/mysql-client                         Run a MySQL client from a docker container      20                  [OK]
fradelg/mysql-cron-backup                 MySQL/MariaDB database backup using cron tas…   18                  [OK]
ansibleplaybookbundle/mysql-apb           An APB which deploys RHSCL MySQL                3                   [OK]
idoall/mysql                              MySQL is a widely used, open-source relation…   3                   [OK]
widdpim/mysql-client                      Dockerized MySQL Client (5.7) including Curl…   1                   [OK]
$
```

docker pull

Deze opdracht wordt gebruikt om een afbeelding of een externe opslagplaats uit een register op te halen. Er zijn veel vooraf gebouwde afbeeldingen die u kunt gebruiken zonder dat u uw eigen afbeeldingen hoeft te definiëren en configureren. De onderstaande opdracht haalt debian image uit dockerhub.

docker pull debian

```
$ docker pull debian
Using default tag: latest
latest: Pulling from library/debian
0e29546d541c: Pull complete
Digest: sha256:2906804d2a64e8a13a434a1a127fe3f6a28bf7cf3696be4223b06276f32f1f2d
Status: Downloaded newer image for debian:latest
$
```

Zoals u kunt zien in de bovenstaande afbeelding, gebruikt de docker-engine standaard de tag :latest. Als u een specifieke versie van de afbeelding moet