PYTHON VERVOLG

CoderDojo

LIJSTEN

Kaart 1 van 6

Ik leer: Python

Geproduceerd door CoderDojo[kennemerwaard]

1 Wat als je een heleboel informatie wilt opslaan? Bijvoorbeeld alle namen van iedereen die bij jouw dojo's aanwezig is? Hier gebruik je een **lijst** voor:

```
list_of_names = ["Alice", "Bob", "Chris"]
list_of_numbers = [1, 2, 3, 4, 5]
```

2 **Lijsten** zijn heel handig. Om te beginnen is het makkelijk om de hele **lijst** te printen.

```
names = [ "Chris", "Bob", "Emma", "Danny"]
print (names)
```

3 Wat als je wilt tellen hoeveel items een **lijst** heeft? Da's makkelijk! Gebruik de **len()** functie. Len is een afkorting voor length (lengte). Pas de code aan en probeer dit:

```
names = ["Chris", "Alice", "Bob", "Emma", "Danny"]
names_count = len(names)
print ("Er zijn "+str(names_count)+)" namen.")
```

4 Gaaf! Maar wat als je de eerste naam van de **lijst** wilt weten? Je gebruikt vierkante haken na de variabele naam, met de positie van het item dat je wilt binnen de haakjes. **Let op, de computer telt vanaf 0!**

```
print ("De eerste naam is "+ names[0]+".")
```

CoderDojo

LIJSTEN

Kaart 1 van 6

Ik leer: Python

Geproduceerd door CoderDojo[kennemerwaard]

5 Je kunt een kopie van een **lijst** maken (dus dan kun je er één aanpassen zonder de ander aan te passen):

```
copy_of_names = list(names)
```

6 Wat als er een nieuwe naam toegevoegd moet worden? Je kunt dat doen met de **append** (voeg toe) methode. Toevoegen betekent eigenlijk "aan het eind erbij plakken". Het werkt als volgt:

```
names.append("George")
```

Probeer een paar namen toe te voegen aan je **lijst**. Gebruik namen van mensen om je heen.

7 Oké, en hoe haal je een naam nu weer van de lijst af? Weer maakt Pyhton dat vrij makkelijk voor je. Je gebruikt de **remove** (verwijder) methode zo:

```
names.remove("Bob")
```

8 Tenslotte is het ook nog mogelijk om de volgorde van de lijst helemaal om te keren door de **reverse** (omdraaien) methode:

```
names.reverse()
```

PYTHON VERVOLG

CoderDojo

FOR LUSSEN

Kaart 2 van 6

Ik leer: Python

Geproduceerd door CoderDojo[kennemerwaard]

1 Deze kaart leert je hoe je **for** loops (voor lussen) moet maken. Deze zijn belangrijk als je iets een bepaald aantal keren wilt doen of voor alles op de lijst wilt gebruiken. Je kunt dit gebruiken om een countdown (aftelprocedure) te **printen**, of alle berichten in een chatgesprek te tonen. Deze kaart geeft een aantal voorbeelden die je kunt maken als programma's om het uit te proberen.

2 Dit programma **print** de nummers 1 tot 10. We doen dit met behulp van de **range** (bereik) functie, die ons alle integers geeft tussen de twee die we in de code schrijven. Het is inclusief het eerste getal, maar stopt voor het laatste getal. De naam die je gebruikt voor "huidige waarde" **variabele** in een **for** loop is niet heel erg belangrijk. Als we alleen maar op deze manier tellen, gebruiken programmeurs vaak de letter **i** die staat voor **integer**: het soort getal dat je telt. Hier hebben we iets leukers gedaan.

```
for hamster in range (1, 11):
    print (str(hamster))
```

Deze loop werkt op de volgende manier:

- Aan het begin van de loop, gebruik het eerste getal dat nog niet door de loop gebruikt is (bij de eerste keer uitvoeren is dat 1, bij de tweede keer 2)
- Sla dat nummer op in de variabele genaamd **hamster**
- Terwijl de rest van de loop uitgevoerd wordt, heeft **hamster** die waarde
- Zodra het programma afgelopen is, voer dan de loop opnieuw uit, tenzij er geen getallen meer zijn

CoderDojo

FOR LUSSEN

Kaart 2 van 6

Ik leer: Python

Geproduceerd door CoderDojo[kennemerwaard]

3 De volgende loop zegt hallo tegen iedereen die op de **lijst** met namen staat. Met de juiste code, kun je deze boodschap gebruiken voor iedereen die jij kent!

```
for name in names:
    print ("Hallo "+name+"!")
```

Deze loop werkt op de volgende manier:

- Aan het begin van de loop, gebruik de eerste naam die nog niet door de loop gebruikt is
- Sla die naam op in de variabele genaamd **name**
- Terwijl de rest van de loop uitgevoerd wordt heeft **name** die waarde
- Zodra het programma afgelopen is, voer dan de loop opnieuw uit, tenzij er geen namen meer zijn.

PYTHON VERVOLG

CoderDojo

WOORDENBOEKEN

Kaart 3 van 6

Ik leer: Python

Geproduceerd door CoderDojo[kennemerwaard]

1 **Lijsten** zijn erg handig, maar sommige dingen zijn erg lastig om daarmee te doen. Laten we wat informatie **pets** (huisdieren) houden. Je kunt dit zo doen:

```
pet_names = ["Fluffy", "Spot", "Felix"]
pet_type = ["konijn", "hond", "kat"]
pet_size = ["klein", "medium", "groot"]
pet_eats = ["wortelen", "botten", "vis"]
```

Maar nu is het lastig om dingen toe te voegen en je kunt ze niet **sorteren**, omdat er geen verband is tussen de eigenschappen en de huisdieren.

2 We kunnen dit oplossen door een nieuw soort **variabele** te gebruiken, die veel met de **lijst** te maken heeft: de **dictionary** (woordenboek). Dan kun je de **waarden** in de dictionary opzoeken met de **key** (sleutel). Probeer dit uit in je code!

```
fluffy = {
    "name" : "Fluffy",
    "type" : "konijn",
    "size" : "klein",
    "eats" : "wortelen"
}
# Doe hetzelfde voor Spot en Felix
# We kunnen informatie krijgen door de juiste sleutel
te gebruiken.
print (fluffy["name"]+" is een "+fluffy["size"]+"
"+fluffy["type"]+" dat graag "+fluffy["eats"]+"eet.")
```

CoderDojo

WOORDENBOEKEN

Kaart 3 van 6

Ik leer: Python

Geproduceerd door CoderDojo[kennemerwaard]

3

Nu zul je denken: "als ik al deze huisdieren moet printen, dan blijf ik maar hetzelfde doen. ik zou een **for loop** moeten kunnen gebruiken!" Dat kan, maar dan moet je de **dictionary** ín een **dictionary** stoppen (of in een **lijst**):

```
# Maak een lege dictionary
pets = ()
# Een huisdier aan de dictionary toevoegen is simpel
pets["Fluffy"] = {
    "name" : "Fluffy",
    "type" : "konijn",
    "size" : "klein",
    "eats" : "wortelen"
}
pets["Spot"] = {
    "name" : "Spot",
    "type" : "hond",
    "size" : "groot",
    "eats" : "botten"
}
pets["Felix"] = {
    "name" : "Felix",
    "type" : "kat",
    "size" : "medium",
    "eats" : "vis"
}
# Nu kun je deze huisdieren doorlopen met een loop.
# Let op: je gebruikt steeds twee paar vierkante
haken.
for pets in pets:
    print (pets[pet]["name"]+" is een "+pets[pet]
["size"]+" "
    +pets[pet]["type"]+" die graag "+pets[pet]
["eats"]"eet.")
```

PYTHON VERVOLG

CoderDojo

FUNCTIES

Kaart 4 van 6

Ik leer: Python

Geproduceerd door CoderDojo[kennemerwaard]

1 Je hebt op de voorgaande kaarten gezien hoe programmeurs loops gebruiken om te voorkomen dat ze dezelfde code steeds opnieuw moeten schrijven. Er is nog een manier om dezelfde code te hergebruiken: je kunt **functions** (functies) schrijven.

Functies laten je een label op een stukje code plakken en het "aanroepen" door de naam van dat label te gebruiken op een andere plaats in je programma. **Print** is een **functie** en dat geldt ook voor de **list functions** (lijst functies) voor sorteren, toevoegen en verwijderen van items.

2 Je creëert een functie door het woord **def** (definitie) te gebruiken. Als je een functie wilt maken die hallo tegen mensen zegt, kun je dat zo doen:

```
def greet(name):
    print ("Hallo "+name+"!")
```

Deze **functie** heet "greet" (groet) en je moet het een **variabele** meegeven die **name** (naam) heet als je het aanroept. Hier een voorbeeld van hoe je het kunt gebruiken:

```
def greet(name):
    print ("Hallo "+name+"!")
# De hoeveelheid code maakt niet uit, maar de functie
moet boven de plaats staan waar je het aanroept.
greet("Bob")
```

Functies zijn met name handig wanneer je hetzelfde wilt doen op verschillende plekken in je code.

CoderDojo

FUNCTIES

Kaart 4 van 6

Ik leer: Python

Geproduceerd door CoderDojo[kennemerwaard]

3

Je kunt veel meer doen met een **functie** dan alleen een beetje code versnellen. Hier is een nuttig voorbeeld, waar we een "add_person" (voeg persoon toe) **functie** gebruiken. Probeer deze code uit te voeren en te gebruiken.

```
people = []
# Creëer een functie om mensen op de lijst te zetten
def get_person():
    # Verzamel informatie over die persoon
    person_name = input("Vul de naam in:")
    person_age = int(input("Vul de leeftijd in") )
    # Wijzig die informatie in een dictionary
    person = {
        "name" : person_name,
        "age" : person_age
    }
    # Geef de persoon terug
    return person

# Hier een kort programma dat het gebruikt
person_count = int(input("Hoeveel personen
toevoegen?"))
for count in range (person_count):
    people.append (get_person())
print ("Je hebt "+str(person_count)+"personen
toegevoegd!")
print ("Dit zijn ze: "+str(people))
```

Deze **functie** neemt geen variabelen als argument aan, maar stelt een paar vragen en **geeft** dan het resultaat **terug** als een "person" dictionary welke **toegevoegd** is aan een **lijst** met personen.

PYTHON VERVOLG

CoderDojo

BESTANDEN

Kaart 5 van 6

Ik leer: Python

Geproduceerd door CoderDojo[kennemerwaard]

1 Tot nu toe hebben je programma's alleen input gekregen van gebruikers door ze rechtstreeks vragen te stellen. Je hebt ook nog niets kunnen opslaan. Dat ga je op deze kaart leren! Je kunt bestanden gebruiken om data te **lezen** en te **schrijven** vanuit je programma, en bijvoorbeeld on een lijst met favoriete huisdieren te bewaren buiten het programma.

2 Als eerste heb je een bestand nodig om mee te werken. Ga naar [dojo.soy/py-file] (http://dojo.soy/py-file) en bewaar dat bestand op dezelfde plek op je computer als je Python programma.

3 Zodra je het bestand hebt, kun je het in je programma openen. Probeer dit uit om de inhoud van het bestand te lezen en te printen zonder de code te veranderen:

```
# Creëer een variabele die het bestand bevat.
# De "r" betekent dat je vanuit het bestand leest
# Je gebruikt een "w" om er naar te schrijven
file = open("python-exmaple.txt", "r")
file_text = file.read()
print (file_text)
```

CoderDojo

BESTANDEN

Kaart 5 van 6

Ik leer: Python

Geproduceerd door CoderDojo[kennemerwaard]

4 Als je naar een bestand schrijft kun je dat op twee manieren doen:

- Open een nieuw bestand met die naam (dit overschrijft bestaande bestanden) en schrijf. Dit is standaard **schrijven** en we gebruiken de **w** in de **open functie**. Je kunt dit doen als je óf de bestaande inhoud wilt wijzigen óf een nieuw bestand wilt maken.
- Open een bestaand bestand met die naam (of maak een bestand met die naam aan als het nog niet bestaat) en schrijf. Dit heet **toevoegen** en we gebruiken een **a** (appending) in de **open functie**. Dit kan handig zijn als je iets wilt toevoegen aan het eind van een bestaand bestand. Je zou het kunnen gebruiken om een backup te maken van een chatgesprek. Probeer beide manieren uit om te zien hoe dit werkt, en open elke keer het **output-example.txt** bestand om het resultaat te bekijken.

Schrijven

```
file = open("output-example.txt", "w")
# Je kunt natuurlijk ook een variabele voor de
bestandsnaam gebruiken, of de output tekst
file.write("Hallo allemaal. Nu kan ik naar een bestand
schrijven!")
```

Toevoegen

```
file = open("output-example.txt", "a")
# Merk op dat deze tekst begint met "\n".
# Dit is om er zeker van te zijn dat datgene wat je
toevoegt op een nieuwe regel begint.
# Dat werkt alsof je overal waar die '\n' in de tekst
staat op de enter toets hebt gedrukt.
file.write("\nHallo allemaal. Nu kan ik naar bestanden
schrijven!")
```

PYTHON VERVOLG

CoderDojo

UITDAGING!

Kaart 6 van 6

Ik leer: Python

Geproduceerd door CoderDojo[kennemerwaard]

1 Dus nu heb je geleerd over:

- **lijsten**: lijsten van variabelen die makkelijker te sorteren, ordenen en te herhalen zijn
- **dictionaries**: een collectie variabelen met makkelijk te gebruiken labels
- **for** loops: herhaal een code een bepaald aantal keer, of loop een lijst langs
- **functies**: hergebruik je code, of die van iemand anders
- Bestanden: lees eruit, schrijf er naar, en voeg iets toe aan een tekstbestand

2 Ga naar dojo.soy/py-pal en bewaar dat bestand op je computer. Je gaat een programma schrijven dat:

- de inhoud **leest** van het bestand dat je gedownload hebt
- kijkt naar wat op elke regel staat en controleert **of** de regel een **palindroom** (keerwoord) is (een woord dat van voor naar achter én van achter naar voor hetzelfde gespeld wordt). Bijvoorbeeld "parterretrap"
- de regels **schrijft** naar twee verschillende bestanden: de regels die palindromen zijn en de regels die dat niet zijn. Hier wat dingen die je moet weten:

3 Verdeel een **bestand** in regels en **print** ze. De **rstrip("\r\n")** functie verwijdert de nieuwe regel ('enter') instructies aan het eind van elke regel.

```
my_file = open('py-pal.txt', 'r')
file_lines = my_file.readlines()
for line in file_lines:
    print (line.rstrip("\r\n"))
```

PYTHON VERVOLG

CoderDojo

UITDAGING!

Kaart 6 van 6

Ik leer: Python

Geproduceerd door CoderDojo[kennemerwaard]

4 Zorg dat alle letters in een **string** even groot zijn (kleine letters of HOOFDLETTERS), omdat Python **a** en **A** niet als dezelfde letter ziet.

```
my_text = 'Hallo ALLEmaal'
lower_my_text = my_text.lower()
upper_my_text = my_text.upper()
```

5 Controleer of een item wel of niet in een lijst staat

```
my_pets = ['cat', 'dog', 'rabbit']

if ('rabbit' in my_pets):
    print ("Ik heb een konijn!")

if ('snake' not in my_pets):
    print ("I heb geen slang.")
```

6 Verander een **string** in een **lijst** met de **karakters** waaruit de string bestaat.

```
my_string = "Hello world!"
my_list = list(my_string)
```

7 Nu heb je alle ingrediënten om deze puzzel op te lossen. Het is geen makkelijke opdracht, dus je kunt mijn oplossing vinden op [dojo.soy/py2ans] (http://dojo.soy/py2ans) als je vast komt te zitten. Als je klaar bent, laat ons weten wat je van deze kaartenreeks vond via dojo.soy/py-intermediate!