RUBY VOOR BEGINNERS

CoderDojo

RUBY INSTALLEREN

Kaart 1 van 7
Ik leer: Ruby

Geproduceerd door CoderDojo[kennemerwaard]

1 Je gaat leren programmeren in Ruby. En daarna in Ruby een spelletje maken.

Wist je dat?

Websites zoals Twitch en Twitter gemaakt zijn met behulp van Ruby?

Voordat je begint met programmeren, moet je Ruby installeren. Het is ook handig om een tekst bewerkingsprogramma te hebben.

2 Om Ruby te installeren ga je naar railsinstaller.org en klik je op de **Windows Ruby 2.3** knop.

Start de installer en kies overal de standaard opties.

3 Nu is het handig om een tekst bewerkingsprogramma te installeren. We raden Atom aan, die je kan downloaden van http:://atom.io, maar notepad werkt ook als je dat liever hebt.

CoderDojo

RUBY INSTALLEREN

Kaart 1 van 7
Ik leer: Ruby

Geproduceerd door CoderDojo[kennemerwaard]

4 Zodra je Ruby en je tekst bewerkingsprogramma hebt geïnstalleerd, gaan we controleren of alles werkt.

- Maak een nieuwe map voor je Ruby Sushi Kaart projecten
- Open je tekst bewerkingsprogramma en maak een nieuw bestand aan. Bewaar het bestand in de map die je net hebt aangemaakt, en noem het **beginner_sushi.rb**
- Open de opdracht prompt (dit heet **command prompt** op Windows en **Terminal** op een Mac) en navigeer naar je map met het **cd** commando.
- Zodra je in de map zit, kan je testen of je Ruby programma werkt. Dit doe je door het volgende in te typen, gevolgd met een enter.

```
ruby beginner_sushi.rb
```

Het is gelukt wanneer je geen meldingen ziet in de opdracht prompt.

Een programma draaien

In volgende kaarten word je gevraagd om een programma te draaien. Dit betekent wat je net hebt gedaan. Naar de map navigeren waar je programma staat en **ruby** in te typen gevolgd door de naam van je programma.

RUBY VOOR BEGINNERS

CoderDojo

JE EERSTE RUBY PROGRAMMA

Kaart 2 van 7
Ik leer: Ruby

Geproduceerd door CoderDojo[kennemerwaard]

1 Tijd voor je eerste Ruby programma. Je gaat de computer hallo laten zeggen tegen iedereen. Type het volgende in je bestand:

```
puts("Hallo iedereen")
```

Draai dit programma en kijk wat er gebeurt! Verander wat er tussen de " symbolen staat, bijvoorbeeld je naam, en draai je programma weer.

2 Voeg nu nog een regel toe. Probeer je code hierop te laten lijken:

```
puts("Hallo iedereen")
puts("Wat is programmeren leuk he?")
```

Draai je programma weer. Zie je hoe de tekst van de tweede puts (genaamd een **string**) op een nieuwe regel begint? Dit komt omdat de computer de volgende opdrachten krijgt:

- Bedenk welke tekst er tussen de ronde haakjes staat
- Schrijf de tekst op het scherm
- Ga naar de volgende regel

CoderDojo

JE EERSTE RUBY PROGRAMMA

Kaart 2 van 7
Ik leer: Ruby

Geproduceerd door CoderDojo[kennemerwaard]

3 Waarom moet de computer bedenken welke tekst er tussen de ronde haakjes staat? Dat komt omdat je tekst uit meerdere **strings** kan bestaan. Probeer het maar! Gebruik onderstaande code, maar vul je eigen naam in waar "mijn naam" staat. (Wel de " blijven gebruiken!):

```
naam = "mijn naam"
puts("Hallo "+naam)
puts("Wat is programmeren leuk he?")
```

De spatie na "Hallo"

Je moet een spatie toevoegen na Hallo, omdat je anders "Hallomijn naam" te zien krijgt.

4 Je hebt net een **variabele** aangemaakt, genaamd **naam**. Dit is alsof je een doos in de computer hebt met een label erop. Je mag alles in die doos doen en dat kan je dan later ophalen door de label te gebruiken. Jij hebt net de **variabele naam** aangemaakt en daar heb je "mijn naam" in bewaard. Op de volgende regel heb je die variabele gebruikt om hallo tegen jezelf te zeggen. Dit heb je gedaan door **naam** aan de **string** te plakken met het **+** symbool.

RUBY VOOR BEGINNERS

CoderDojo

PRATEN MET DE GEBUIKER

Kaart 3 van 7
Ik leer: Ruby

Geproduceerd door CoderDojo[kennemerwaard]

1 Het is wel leuk om de computer je naam aan het einde van **"Hallo "** te plakken, maar waarom schrijf je niet gewoon **"Hallo [mijn naam]"**? Dat is omdat je met een **variabele** niet hoeft te weten wat voor waarde die krijgt om je programma te schrijven. Je kan zelfs de gebruiker van je programma om een naam vragen. Wijzig je Ruby programma om dat te doen:

```
puts("Hoe heet jij?")
naam = gets.chomp
puts("Hallo "+naam)
puts("Wat is programmeren leuk he?")
```

Probeer je programma maar te draaien. Je moet wel op "Enter" drukken zodra je jouw naam hebt ingevuld.

2 We gaan nu proberen een getal aan de gebruiker te vragen. Zie je dat je **+** aan beide kanten van de variabele kunt gebruiken?

Draai dit programma en beantwoord de vragen. Zie je wat er gebeurt?

```
puts("Hoe heet jij?")
naam = gets.chomp
puts("Hallo "+naam+", kies een getal")
mijn_getal = gets.chomp
puts("Jouw getal is "+mijn_getal)
```

CoderDojo

PRATEN MET DE GEBUIKER

Kaart 3 van 7
Ik leer: Ruby

Geproduceerd door CoderDojo[kennemerwaard]

3 Wat als je nu een getal wilt optellen bij je variabele? Voeg een regel code toe aan je programma dat 1 optelt bij je variabele.

```
puts("Hoe heet jij?")
naam = gets.chomp
puts("Hallo "+naam+", kies een getal")
mijn_getal = gets.chomp
mijn_getal = mijn_getal.to_i + 1
puts("Jouw getal is "+mijn_getal.to_s)
```

Zie je hoe een waarde uit een variabele hebt gepakt, opgehoogd en weer opgeslagen in die variabele? En dat allemaal op dezelfde regel!

Maar waarom moesten we **.to_i** en **.to_s** aanroepn?

Dat komt omdat Ruby het getal '1' anders ziet als je ermee wilt rekenen, dan wanneer je het in een zin gebruikt. Als je **.to_i** aanroept op een variabele, maakt Ruby er een **integer** (wiskundig getal) van. Als je er **.to_s** op aanroept, maakt Ruby er een **string** (woord) van.

integers en **strings** zijn variabele **types** en sommige stukken code (zoals **+** en **puts**) werken alleen als je variabelen de juiste types hebben.

Wiskunde

Je hebt gezien hoe je moet optellen, maar je kan ook:

- Aftrekken met **-**
- Vermenigvuldigen met *
- Delen met **/**

RUBY VOOR BEGINNERS

CoderDojo

VERGELIJKINGEN EN 'IF CONDITIES'

Kaart 4 van 7
Ik leer: Ruby

Geproduceerd door CoderDojo[kennemerwaard]

1 Je kan aan Ruby vragen twee getallen met elkaar te vergelijken. Dit kan erg handig zijn (kost die broek meer dan je in je spaarpot hebt?) Dit kan je doen met speciale symbolen:

- **a > b** vraagt of **a** groter is dan **b**
- **a < b** vraagt of **a** kleiner is dan **b**
- **a == b** vraagt of **a** hetzelfde is als **b**
- **a != b** vraagt of **a** ongelijk is aan **b**
- **a >= b** vraagt of **a** groter of gelijk is aan **b**
- **a <= b** vraagt of **a** kleiner of gelijk is aan **b**

==

Het dubbele is-gelijk **==** teken wordt gebruikt om variabelen te **vergelijken**. Want een losse **=** wordt al gebruikt om een variabele een waarde te geven.

2 Je kunt deze vergelijkingen in een **if** (als-dan) structuur gebruiken. Code in een **if** structuur wordt alleen uitgevoerd als de voorwaarde tussen de haakjes waar is. In dit voorbeeld wordt er tekst getoond.

```
if(mijn_getal > 100)
  puts("Dat is een groot getal!")
end
```

Inspringing

De **puts** is **ingesprongen**. Dat betekent dat er twee spaties extra voor staan. Dit is gedaan om je code leesbaarder te maken. Ruby zal ook werken zonder die spaties.

CoderDojo

VERGELIJKINGEN EN 'IF CONDITIES'

Kaart 4 van 7
Ik leer: Ruby

Geproduceerd door CoderDojo[kennemerwaard]

3 Laten we deze code nu toevoegen aan je programma van de vorige kaart. Verander je programma zodat het er zo uitziet:

```
puts("Hoe heet jij?")
naam = gets.chomp
puts("Hallo "+naam+", kies een getal")
mijn_getal = gets.chomp
mijn_getal = mijn_getal.to_i

if(mijn_getal > 100)
  puts("Dat is een groot getal!")
end
```

Probeer je nieuwe programma uit met verschillende getallen. Probeer getallen kleiner en groter dan 100 en kijk wat er gebeurt. Wat zou er gebeuren als je precies 100 invult?

4 Je kunt ook voorwaarden met elkaar combineren, door **and** en **or** ("en" en "of") te gebruiken. Zo kun je bijvoorbeeld deze code schrijven:

```
if(mijn_getal >= 20 and mijn_getal < 30)
  puts("Dat getal is ergens in de 20!")
end
```

Of bijvoorbeeld:

```
 if(eten == "taart" or eten == "chocolade" or eten ==
"snoep")
   puts("Klinkt erg lekker!")
 end
```

CoderDojo

BIJZONDERE IF CONDITIES

Kaart 5 van 7
Ik leer: Ruby

Geproduceerd door CoderDojo[kennemerwaard]

1 Wat als je wilt controleren of het getal hoog genoeg is? Bijvoorbeeld als de gebruiker een getal hoger dan 100 invult. Vertel dan dat het getal hoog genoeg is. Of vertel de gebruiker dat het getal te laag is. Probeer het volgende:

```
 puts("Hoe heet jij?")
 naam = gets.chomp
 puts("Hallo "+naam+", kies een getal dat hoger is dan
100")
 mijn_getal = gets.chomp
 mijn_getal = mijn_getal.to_i
 puts("Jouw getal is " + mijn_getal.to_s)

if(mijn_getal > 100)
  puts("Dat getal is hoog genoeg!")
else
  puts("Dat getal is te laag!")
end
```

De **else** werkt als een "anders". Hier staat dus eigenlijk "**als** mijn getal groter is dan 100 zeg dan dat het hoog genoeg is en **anders** zeg je dat het getal te laag is"

RUBY VOOR BEGINNERS

CoderDojo

BIJZONDERE IF CONDITIES

Kaart 5 van 7
Ik leer: Ruby

Geproduceerd door CoderDojo[kennemerwaard]

2

En wat als we de gebruiker willen vertellen dat ze wel in de buurt zitten als ze een getal groter dan 90 hebben gekozen? Dan gebruik je **elsif**. Dat is **else** en **if** samengevoegd. Dat betekent **of als**. Dat wordt alleen uitgevoerd als de voorwaarde in de **if** niet waar is en de voorwaarde in de **elsif** wel waar is. Dit moet je toevoegen om te vertellen dat de gebruiker in de buurt is:

```
elsif(mijn_getal > 90):
    print("Je bent er bijna!")
```

En zo ziet het programma er dan in zijn geheel uit. Zie je dat je de **elsif** tussen de **if** en de **else** condities moet zetten?

```
 puts("Hoe heet jij?")
 naam = gets.chomp
 puts("Hallo "+naam+", kies een getal dat hoger is dan
100")
 mijn_getal = gets.chomp
 mijn_getal = mijn_getal.to_i
 puts("Jouw getal is " + mijn_getal.to_s)

 if(mijn_getal > 100)
   puts("Dat getal is hoog genoeg!")
 elsif(mijn_getal > 90)
   puts("Je bent er bijna!")
 else
   puts("Dat getal is te laag!")
 end
```

RUBY VOOR BEGINNERS

CoderDojo

HERHALINGEN EN 'LOOPS'

Kaart 6 van 7
Ik leer: Ruby

Geproduceerd door CoderDojo[kennemerwaard]

1 Je weet nu hoe je een gebruiker om een getal moet vragen, en als dat getal niet goed is, hoe je dat moet vertellen. Maar wat als je wilt doorgaan totdat de gebruiker het juiste getal heeft? Je kunt dan een **if** structuur in een andere **if** structuur plaatsen. Maar wat als de gebruiker het antwoord dan nog steeds niet heeft? Je hebt een manier nodig om net zo lang te blijven vragen, totdat de gebruiker het juiste getal heeft ingevuld. Om de computer dit te laten vragen gebruik je een **loop** (herhaling). Je gaat hier de **while** (terwijl) herhaling gebruiken.

2 Een **while** herhaling, lijkt een beetje op een **if** conditie: alleen als de voorwaarde waar is wordt de code uitgevoerd. Het verschil is dat een **while** dit blijft doen, totdat de voorwaarde niet meer waar is. Je moet wel zorgen dat je een **while** loop kan stoppen. Anders gaat je programma oneindig lang door! De code ziet er zo uit:

```
while(mijn_getal < 100) do
 puts("Hallo "+naam+", kies een getal dat hoger is dan
100")
 mijn_getal = gets.chomp
 mijn_getal = mijn_getal.to_i
end
```

CoderDojo

HERHALINGEN EN 'LOOPS'

Kaart 6 van 7
Ik leer: Ruby

Geproduceerd door CoderDojo[kennemerwaard]

3 Voeg nu de **while** herhaling toe aan je programma.

```
puts("Hoe heet jij?")
naam = gets.chomp
mijn_getal = 0

# herhaal terwijl "mijn_getal" kleiner of gelijk is
aan 100
while(mijn_getal <= 100) do
 # vraag de gebruiker om een getal
 puts("Hallo "+naam+", kies een getal dat hoger is dan
100")
 mijn_getal = gets.chomp
 # verander het antwoord naar een getal
 mijn_getal = mijn_getal.to_i
 puts("Jouw getal is " + mijn_getal.to_s)

 # controleer of het getal groter is dan 100
 if(mijn_getal > 100)
   puts("Dat getal is hoog genoeg!")
 elsif(mijn_getal > 90)
   puts("Je bent er bijna!")
 else
   puts("Dat getal is te laag!")
 end
end
```

Commentaar

Dit zijn notities voor programmeurs (of jij later) die de computer negeert. In Ruby begint commentaar met een # en geldt het voor de hele regel.

RUBY VOOR BEGINNERS

CoderDojo

TIJD VOOR EEN SPELLETJE!

Kaart 7 van 7
Ik leer: Ruby

Geproduceerd door CoderDojo[kennemerwaard]

1 Je hebt nu dus geleerd over:

- **puts**: Berichten versturen naar een gebruiker
- variabelen: Een manier om waarden te onthouden en bij te werken
- strings: Stukjes tekst
- **input**: Hoe je vragen stelt aan een gebruiker
- wiskunde: Hoe je moet rekenen met getallen
- integers: Getallen om mee te rekenen
- **if** condities: Iets doen als de voorwaarde waar is
- **while** loops: Iets blijven doen totdat je voorwaarde niet meer waar is

2 Probeer het volgende spelletje te maken:

- Er is een getal (integer), van 0 t/m 9, dat je programma in het geheim kiest.
- Een speler mag 5 keer het getal raden
- Het spel legt de regels uit aan de speler
- Na elke poging vertelt het programma of het geheime getal lager, hoger of precies goed is. En hoeveel keer de speler nog mag raden.
- Als de speler het getal goed heeft, feliciteert het programma de speler
- Als de speler bij de 5e poging het getal nog niet heeft geraden is het spel afgelopen en heeft de speler verloren

3 Een voorbeeld van het spel kun je vinden op dojo.soy/py-dice (in het Engels).

CoderDojo

TIJD VOOR EEN SPELLETJE!

Kaart 7 van 7
Ik leer: Ruby

Geproduceerd door CoderDojo[kennemerwaard]

4 Je mist nog nog één ding om dit spel te maken: Een manier om een willekeurig getal van 0 t/m 9 te maken. De code hiervoor is nu nog misschien een beetje lastig. Maar dat leggen we bij de volgende Sushi Kaarten wel uit! Gebruik de volgende regel code om een geheim getal tussen 0 en 9 op te slaan in een variabele:

```
geheim_getal = (rand * 9).to_i
```

5 Probeer nu het spel te maken! Vergeet de vorige Sushi Kaarten niet te gebruiken. Als je niet verder komt of als je klaar bent, kun je hier een antwoord vinden voorbeeld.rb. Maak je geen zorgen als jouw programma er anders uit ziet. Als het maar werkt. Succes!