

ECTTP: Dictionaries

Valentijn Muijers
<https://github.com/vmuijers/ECTTP>

•

•

Course Overview

- Week One: Course overview
- Week One: Variables
- Week Two: Conditions
- Week Three: Loops
- Week Four: Functions
- Week Five: Tuples
- Week Six: Lists
- Week Seven: How to think as a programmer
- First Test! (vrijdag 21 oktober)
- Week Eight: Classes and Objects
- **Week Nine: Overview ←**
- Week Ten: Alleen werkcolleges
- Week Eleven: Alleen werkcolleges
- **Second Test!**

Classes

- Blauwdruk van een object
- Bevat eigenschappen van het object
- (dit zijn variabelen en functies die bij het object horen)
- Voorbeeld:
 - Class Car
 - 4 wielen
 - 1 stuur
 - 4 deuren
 - Kan rijden

Attributen

- **Naam**
 - "Sardauker Trooper"
- **Prijs**
 - Getal, (geldbedrag)
- **Positie_x** en **Positie_y**
 - Getallen (scherm-coordinaten)
- **Health**
 - Getal tussen 0 en 100



De attributen zijn de eigenschappen (variabelen) die de Trooper heeft

Interface

- Class Trooper

Attributen	Methoden
Naam	Attack(x,y)
Prijs	Move(x,y)
Positie_x	Retreat()
Positie_y	Guard(building)
Health	

Attributen en Methoden van de Trooper uit Dune2.

Een Interface beschrijft hoe een class gebruikt kan worden.

De Interface is tevens de documentatie van de class.

In het voorbeeld staan een aantal attributen, mogelijk zijn dit er meer (denk aan Damage, MoveSpeed etc.).

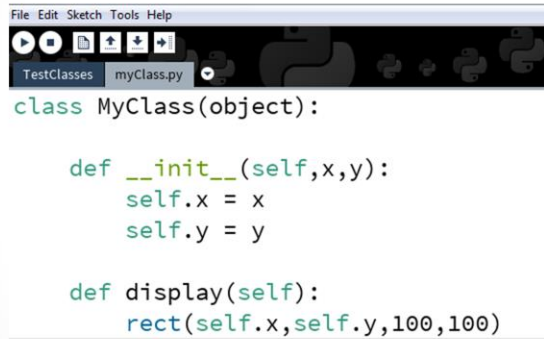
Constructor

`someCar = Car(20, 30, 10)` ← this calls the constructor
of the Car-class

`someCar.x += 10` ← we can now access the attributes
and methods of the car

```
class Car(object):  
    def __init__(self, x, y, speed): ← constructor  
        self.x = x  
        self.y = y  
        self.speed = speed
```

Processing Class



The screenshot shows the Processing IDE interface. The menu bar at the top includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu is a toolbar with icons for running, saving, and other functions. The file explorer shows two tabs: 'TestClasses' and 'myClass.py'. The main code editor displays the following Python code:

```
class MyClass(object):  
  
    def __init__(self,x,y):  
        self.x = x  
        self.y = y  
  
    def display(self):  
        rect(self.x,self.y,100,100)
```

Processing Voorbeeld!



```
from myClass import MyClass
```

```
def setup():  
    size(640, 360)  
    global r  
    r = MyClass(100,100)
```

```
def draw():  
    background(0)  
    r.display()
```


Dictionaries

- Manier om meerdere elementen op te slaan (zoals bij een list)
- Sneller dan een list met opvragen
- Ongeordend
- Een dictionary werkt met Key-Value paren
- De Key is uniek en wordt gebruikt om de Value op te vragen

•

•

Example

- Definitie:
- `phonebook = {}`
- Elementen:
- `phonebook["John"] = 938477566`
- `phonebook["Jack"] = 938377264`
- `phonebook["Jill"] = 947662781`

•

•

Initializing a Dictionary

- phonebook =
- {
- "John" : 938477566,
- "Jack" : 938377264,
- "Jill" : 947662781
- }

•

•

Iterating over dictionary

- `for name, number in phonebook.items():`
 - `print "Phone number of %s is %d" % (name, number)`
 - `print "Phone number "+name+" is " + str(number)`

Removing elements

- `del phonebook["John"]`
- Of
- `phonebook.pop("John")`

•

•

Toets 2

- Maak een programma (met een aantal eisen)
- Basic, Advanced of Expert
- Hoe complexer hoe hoger je cijfer

The Future!....?

- Arduino (C)
- Unity (C#)

Opdracht

- Bedenk welke elementen van programmeren je nog lastig vindt
- Maak een programma waarmee je precies die elementen oefent

Learn to Code!

- <https://www.sololearn.com/Courses/>

Nineth lab is online

https://github.com/vmuijters/ECTP/blob/master/Labs/Lab_9.md

#For examples/tutorials and references!
py.processing.org

#For more practise with python!
codecademy.com

#Now let's practise some more with codingbat:
<http://codingbat.com/python>

•

•