



HBO-ICT: TICT-V1PROG-15

Programming

Werkboek



Inhoudsopgave

1	Week één: Starten met Python	7
1.1	Vorbereiding	7
1.1.1	Opdracht: Python Installeren.....	7
1.1.2	Opdracht: IDE installeren.....	7
1.1.3	Opdracht: De PyCharm Tutorial.....	8
1.1.4	Windows Linux Mac	9
1.2	College 1.....	10
1.2.1	Opdracht: Vorbereiding voor je eerste Py-Charm project.....	10
1.2.2	Opdracht: Arithmetic operators, Bereken het maximale aantal reizen.....	10
1.2.3	Opdracht: Arithmetic operators, Aantal reizigers.....	10
1.2.4	Opdracht: Comparison operators, Vergelijk 3 variabelen met een vaste waarde en zoek de match. .	10
1.2.5	Opdracht: Comparison operators, Vergelijk 3 variabelen en zoek de verschillen.....	10
1.2.6	Opdracht: Assignment operators, Bereken het aantal reizigers.	11
1.2.7	Opdracht: Strings, Naam reiziger.....	11
1.2.8	Opdracht: Strings, Oefeningen	11
1.2.9	Opdracht: PythonIntroduction	12
1.3	College 2.....	13
1.3.1	Opdracht: Python Introduction.....	13
1.3.2	Opdracht: Lists	13
1.3.3	Opdracht: Turtle-huis.....	13
1.3.4	Opdracht: Maak een PSD:.....	13
1.3.5	Opdracht: Functies.....	13
1.4	Weekopdracht	14
1.4.1	Opdracht: PSD.....	14
1.4.2	Opdracht: Kaartautomaat.....	14
1.4.3	Opdracht: Turtle Graphics	14
1.4.4	Opdracht: Rapportage	15
2	Week twee: Control flow	17
2.1	Vorbereiding	17
2.1.1	Opdracht: Maak de geel gearceerde opdrachten:.....	17
2.1.2	Opdracht: Subscripting en Slicing	17
2.1.3	Opdracht: Invoer en uitvoer	18
2.1.4	Opdracht: Als/Dan	18
2.1.5	Opdracht: For loop.....	18
2.1.6	Opdracht: Sommetjes	18

2.1.7	Opdracht: Sommetjes 2	18
2.1.8	Opdracht: Sommetjes 3	18
2.1.9	Opdracht: Lists	18
2.1.10	Opdracht: Functies	18
2.2	College 1.....	19
2.2.1	Opdracht: Volle trein, als-dan... ..	19
2.2.2	Opdracht: Volle trein, als-dan...anders... ..	19
2.2.3	Opdracht: Volle trein, als-dan...anders-dan... anders... ..	19
2.2.4	Opdracht: Iteratie, Kosten per jaar	19
2.2.5	Opdracht: Iteratie, “While” versus “For”	19
2.2.6	Opdracht: Iteratie, “While” versus “For”	20
2.3	College 2.....	21
2.3.1	Opdracht: FC Utrecht fans	21
2.3.2	Opdracht: Ajax fans.....	21
2.4	Weekopdracht	22
2.4.1	Opdracht: Plan je route	22
2.4.2	Opdracht: Route informatie.....	22
2.4.3	Opdracht: Extra beveiliging van de e-ticket.....	23
3	Week drie: Expressies en documentatie	24
3.1	Vorbereiding	24
3.1.1	Logical Operators: AND OR XOR.....	24
3.1.2	Opdracht: Logical Operators.....	25
3.1.3	Membership Operators: in not in	26
3.1.4	Opdracht: Logical Operators.....	26
3.1.5	Opdracht: Logical Operators.....	26
3.1.6	Identity operators: is is not	26
3.1.7	Meer informatie:.....	27
3.1.8	Docstrings:	27
3.1.9	Opdracht: Docstrings	28
3.1.10	Meer informatie:	28
3.2	College 1.....	29
3.2.1	Opdracht: Exceptions, Stap 1.....	29
3.2.2	Opdracht: Exceptions, Stap 2.....	29
3.2.3	Opdracht: Exceptions, Stap 3.....	29
3.2.4	Opdracht: Doctest.....	29
3.3	College 2.....	30
3.3.1	Opdracht 1: Tuples en Dictionaries.....	30

3.3.2	Opdracht: Dictionaries en stationsvoorzieningen.....	30
3.3.3	Opdracht: Functie die een tuple teruggeeft.	30
3.3.4	Opdracht: Sets.....	30
3.3.5	Opdracht: Dictionary in dictionary.....	31
3.4	Weekopdracht	32
3.4.1	Opdracht: Beschrijving.....	32
3.4.2	Opdracht: Vereisten.....	32
3.5	Extra Uitdagende Opdracht: Spoor netwerk	33
3.5.1	Extra Uitdagende Opdracht: Kortste route (minste stations) van Duivendrecht naar Schiphol	33
3.5.2	Extra Uitdagende Opdracht: Voorzieningen per station	33
4	Week vier: Data	34
4.1	Vorbereiding	34
4.1.1	Extra uitdagende opdracht:	34
4.1.2	Opdracht:	34
4.2	College 1.....	35
4.2.1	Opdracht: Een interactieve gebruiker	35
4.2.2	Opdracht: Format je uitvoer	35
4.2.3	Opdracht: Lezen van files.....	35
4.2.4	Opdracht: Schrijven van files	35
4.3	College 2.....	36
4.3.1	Opdracht: Lezen van CSV-bestanden.....	36
4.3.2	Opdracht: Schrijven van CSV-bestanden.	36
4.4	Weekopdracht	37
4.4.1	Extra Uitdagende Opdracht.	37
4.4.2	Opdracht: Maak de onderstaande functionaliteit:.....	38
4.4.3	Hoe moet ik beginnen?.....	39
5	Week vijf: XML.....	40
5.1	Vorbereiding	40
5.1.1	Opdracht: Vind jij de 11 fouten in bijgaand XML document?	40
5.2	College 1.....	41
5.2.1	Opdracht (10 min): API key aanvragen	41
5.2.2	Opdracht: XML.....	41
5.3	College 2.....	42
5.3.1	Opdracht: tkinter	42
5.4	Weekopdracht	43
5.4.1	Casus: Actuele vertrektijden.....	43
5.4.2	Opdracht: De NS API	43

5.4.3	Opdracht: Loop door de treinen en print steeds een enkele trein.....	43
5.4.4	Opdracht: Alleen de vertrektijden van de trein naar “Nijmegen”	44
5.4.5	Opdracht: Andere Eindbestemming.	44
5.4.6	Opdracht: Betere uitvoer.....	44
5.4.7	Opdracht: Tkinter.....	44
5.4.8	Opdracht: Integratie	44
5.5	Extra Uitdagende Opdracht: Giphy API.....	45
5.5.1	Opdracht: Maak een Tkinter scherm waarin je afbeeldingen kunt zoeken en weergeven.....	45
6	Week zes: Object/classes & source control	49
6.1	Vorbereiding	49
6.1.1	"Object/Classes"	49
6.1.2	"Version Control System: Git en Github"	49
6.1.3	Opdracht: Werken met Git	49
6.1.4	Opdracht: Werken met GitHub.....	50
6.2	College 1.....	51
6.2.1	Opdracht: Namespaces.....	51
6.2.2	Opdracht: Object Oriented Programming	51
6.3	Weekopdracht	52
6.3.1	Opdracht: Object-georiënteerd programmeren (PythonIntroduction).....	52
6.3.2	Opdracht: Object-georiënteerd programmeren (OOP)	52
7	Resources	54

Inleiding:

Programmeren kan je alleen maar leren door het heel vaak “te doen”. In dit werkboek vind je alle opdrachten die horen bij deze cursus. Elke les heeft ongeveer dezelfde opbouw:

- **Voorbereiding:** De docent gaat er vanuit dat je deze opdrachten hebt gedaan voordat je naar school gaat om het college bij te wonen.
- **College 1:** Deze opdrachten mag je alvast thuis maken, maar dat hoeft niet: Je maakt deze opdrachten tijdens het college om de lesstof die de docent dan uitlegt gelijk te oefenen tijdens de les. Het is te veel werk om af te krijgen tijdens de les, dus thuis afmaken.
- **College 2:** Zie “College 1”.
- **x.y.z. Opdracht:** Zoals gezegd, je kan alleen maar programmeren leren door het heel veel te doen. Door het maken van de opdracht verwerk je de lesstof van die week, ga je steeds beter programmeren en zorg je ervoor dat je niet zo snel vergeet wat je hebt geleerd. Elke opdracht is genummerd.

Het is de bedoeling dat je de opdracht 1.1 die staat bij **Week 1** uitvoert voordat je op school komt om de lessen te volgen. Dit is dus de voorbereiding van deze cursus.

We gaan er vanuit dat je gebruik kunt maken van een eigen laptop. In beperkte mate kan je ook gebruik maken van de computers op school.

Pas op bij het **kopiëren van code** vanuit dit werkboek naar je IDE (de software die je gebruikt om te programmeren). Een paar tips hierbij zijn:

- 1) Let erop dat er geen spaties aan het einde van de zinnen worden meegekopieerd.
- 2) Let erop dat de aanhalingstekens die je in Word gebruikt: `"""` niet gelijk zijn aan de aanhalingstekens die je met python mag gebruiken: `"""`. Zie je het verschil?
- 3) Dit geldt ook voor: `”` (gebruikt in Word) en `”` (gebruikt in Python).

Soms begint een opdracht met de tekst “**Extra uitdagende opdracht**”. Deze opdracht is bedoeld voor studenten die een hoge score hebben gehaald op hun formatieve toets. Deze opdrachten zijn ook voorzien van een “rode kantlijn” (zie hiernaast).

1 Week één: Starten met Python

1.1 Voorbereiding

Dit is eigenlijk de voorbereiding op deze cursus. We gaan er namelijk vanuit dat je een computer hebt waarop je de programmeeromgeving kunt gaan installeren. Deze omgeving bestaat uit: De programmeertaal “Python” en de zogenaamde Integrated Development Environment (IDE) die het je makkelijker maakt om een python programma te maken. Tijdens les 1 moet je al opdrachten maken en we verwachten dan van jou dat je een laptop hebt waarop Python+IDE is geïnstalleerd...

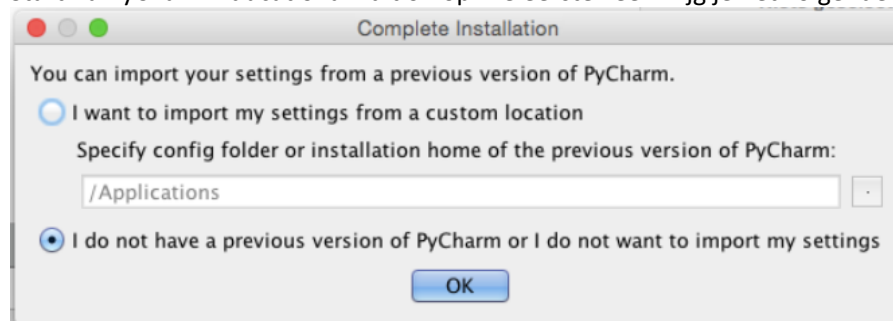
1.1.1 Opdracht: Python Installeren

Ga naar www.python.org. Ga naar Downloads en installeer de laatste versie van Python: Op het moment van schrijven is dit versie 3.4.3. Let op, de versie kan in de loop van het jaar veranderen. Kies voor de release met een hoog nummer beginnend met een “3”. Wij werken met Python 3.x.y.

1.1.2 Opdracht: IDE installeren

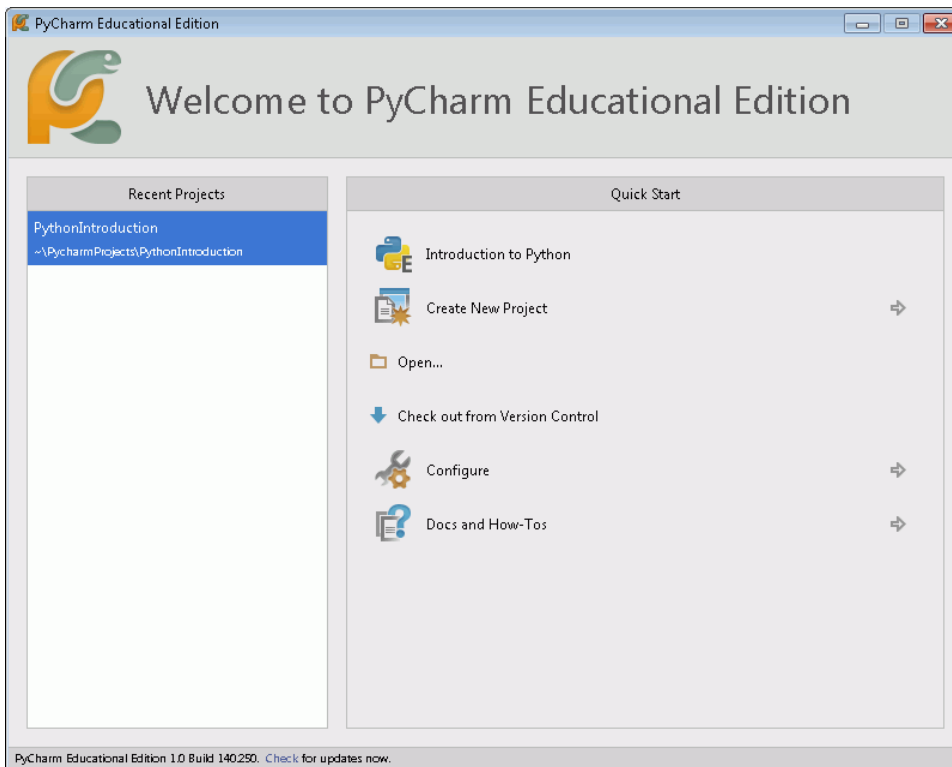
Om Python-programma's te kunnen schrijven, moet je Python-bestanden schrijven en uitvoeren die eindigen op .py. Die moet je vervolgens uitvoeren (middels een compiler), zodat jouw voor-mensen-leesbare-code wordt omgezet naar voor-computers-leesbare-code. Om dat allemaal wat makkelijker te maken, bestaan er Integrated Development Environments (IDE), programma's die je helpen met de code te schrijven in een .py-bestand, maar ze helpen je ook om je code te “compileren”. De IDE die jullie gaan gebruiken, heet PyCharm. We gebruiken de Educational Edition, omdat we daar de opdrachten voor je in hebben klaargezet. Download en installeer PyCharm Educational Edition via <https://www.jetbrains.com/pycharm-educational>.

Start nu PyCharm Educational Edition op. De eerste keer krijg je het volgende venster te zien:



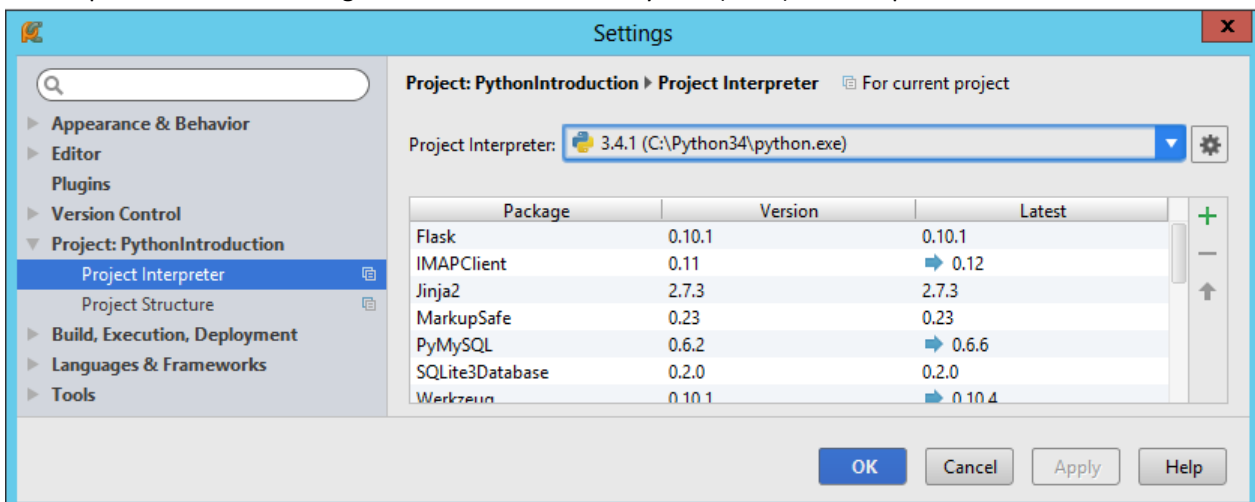
FIGUUR 1: SETTINGS NIET IMPORTEREN.

Waarschijnlijk heb je nog niet eerder in Python geprogrammeerd, dus dan kies je voor de onderste optie. Je ziet nu het startscherm van PyCharm.



FIGUUR 2: PYCHARM INSTALLATIE

We gaan eerst checken of je de juiste versie van Python gebruikt. Klik op Configure en daarna op Settings (Windows). Je krijgt nu een nieuw venster te zien. Klik op Default Project en dan op Project Interpreter. Kies nu in het drop-downmenu voor de geïnstalleerde versie van Python (3.4.3) en klik op OK:



FIGUUR 3: VERSIE VAN PYTHON EN DE NAAM VAN HET PROJECT KUNNEN AFWIJKEN VAN JOUW INSTELLINGEN.

PyCharm voert nu een aantal veranderingen door, waardoor je even moet wachten. Klik nu op de pijl naar links naast Configure om terug te gaan naar het beginscherm.

1.1.3 Opdracht: De PyCharm Tutorial

Je bent nu klaar om je eerste programma te schrijven! Klik hiervoor in het beginscherm op "Introduction to Python" en doorloop de les "Introduction". Het gaat om een paar kleine opdrachtjes.

Klaar? Als je echt lekker bezig bent, dan mag je ook verder gaan met de volgende lessen.

Je bent nu klaar voor de eerste les.

1.1.4 Windows | Linux | Mac

De HU ondersteunt het werken met Python 3 en Pycharm-Educational op het Windows platform. Het is niet onmogelijk en ook niet verboden om deze combinatie te installeren op het Linux platform (bijvoorbeeld Ubuntu) of Mac OS, maar dit wordt niet ondersteund door de opleiding. Als je dit overweegt, lees dan deze pagina eerst door: <https://www.jetbrains.com/pycharm-educational/quickstart/installation.html>

1.2 College 1

1.2.1 Opdracht: Voorbereiding voor je eerste Py-Charm project...

Start Py-Charm en maak een nieuw project aan:

- Ga naar File > New Project
- Kies voor Pure Python
- Vervang “untitled” door de zelfgekozen naam van je project
- Zorg dat er bij “Interpreter” jouw versie van Python staat aangegeven.
- Druk op “Create”
- Kies voor “open in new window”

Een nieuw bestand aanmaken:

- Ga naar File > New...
- Kies voor Python File
- Kies een logische naam
- Druk op OK

1.2.2 Opdracht: Arithmetic operators, Bereken het maximale aantal reizen...

- Roxanne reist dagelijks op en neer tussen Utrecht Centraal en Veenendaal-De Klomp
- Ze heeft 83 euro op haar ov-chipkaart staan
- Op en neer reizen kost 13 euro
- Print hoeveel dagen Roxanne op en neer kan reizen
 - NB: Er is geen minimaal bedrag dat Roxanne op haar ov-chipkaart moet hebben staan om te kunnen reizen. Je kunt geen negatief saldo hebben.

1.2.3 Opdracht: Arithmetic operators, Aantal reizigers...

- Er zitten 0 reizigers in de trein.
- Op station Utrecht Centraal stappen 2 reizigers in de trein
- Op ieder volgend station verdubbelt het aantal reizigers
- Print hoeveel reizigers er bij het 10^e stations in de trein zitten

1.2.4 Opdracht: Comparison operators, Vergelijk 3 variabelen met een vaste waarde en zoek de match.

- De NS heeft de volgende ov-chipkaartnummers in hun database staan:
 - 201401053
 - 201501018
 - 201412227
- Sla ieder nummer op in een afzonderlijke variabele.
- Nu houdt de klant ov-chipkaart 20140101053 voor de kaartautomaat
 - check voor iedere variabele of deze hetzelfde is en als dat zo is, print “Match!”

1.2.5 Opdracht: Comparison operators, Vergelijk 3 variabelen en zoek de verschillen.

- De NS heeft de volgende ov-chipkaartnummers in haar database staan:
 - 201401053

- 201501018
- 201412227
- Sla ieder nummer op in een afzonderlijke variabele.
- de NS heeft een actie: iedere kaart met nummer groter of gelijk aan 201412227 krijgt korting.
 - Check voor iedere variabele of deze groter of gelijk is aan 201412227 en als dat zo is, print “Korting!”.

1.2.6 Opdracht: Assignment operators, Bereken het aantal reizigers.

- Er zitten 0 reizigers in de trein
- Op station Utrecht Centraal stappen 256 reizigers in de trein
- Op station ‘s Hertogenbosch stappen 102 mensen uit
- Op station ‘s Hertogenbosch stappen 84 mensen in
- Op station Eindhoven wordt de trein gesplitst in tweeën, het aantal reizigers wordt precies door twee gedeeld.
- Opdracht: maak een variabele met de naam “aantal” aan, en voer de vier stappen uit met assignment operators en print het eindresultaat.

1.2.7 Opdracht: Strings, Naam reiziger.

- De kaartautomaat geeft nu nog de welkomsttekst “Hallo NS-reiziger!”
- Zorg dat de kaartautomaat deze tekst weergeeft als output op de console.
- Zorg daarna dat de kaartautomaat jouw naam weergeeft in plaats van “NS-reiziger”. Doe dit door “NS-reiziger” te vervangen door “<<jouw naam>>” en daarna te printen.
- Tip: Gebruik “replace”.

1.2.8 Opdracht: Strings, Oefeningen

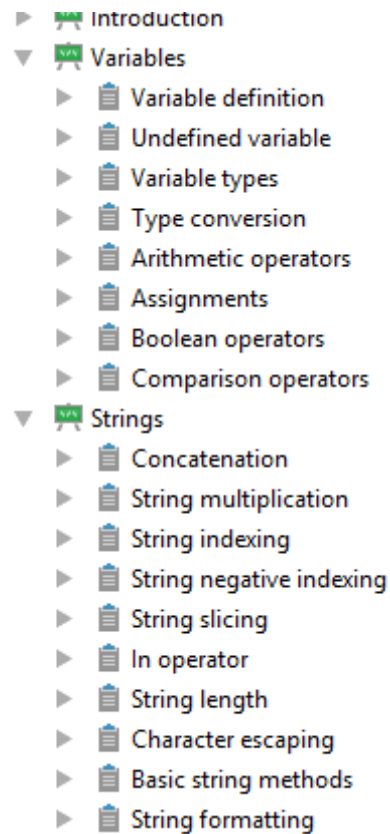
1. Maak string variabele “dit is een tekst” en print deze (zonder aanhalingstekens).
2. print de tekst Dit is: “Bea's Huis” (met aanhalingstekens en apostroph).
3. print karakter 5 uit “hello world” (let op, dat is dus de letter “o”)
4. print karakter 3-7 uit “hello world”
5. print de laatste karakter uit “hello world”
6. print karakter 1 t/m 10 uit “hello world” op twee verschillende manieren.

1.2.9 Opdracht: PythonIntroduction

Heropen het Py-Charm PythonIntroduction project weer (bv via File → Reopen project → PythonIntroduction) en maak de onderstaande oefeningen uit de lessen 1 t/m 4. Dit is tevens de voorbereiding op het volgende college.

Maak de drie lessen:

- “Introduction”,
- “Variables” en
- “Strings”

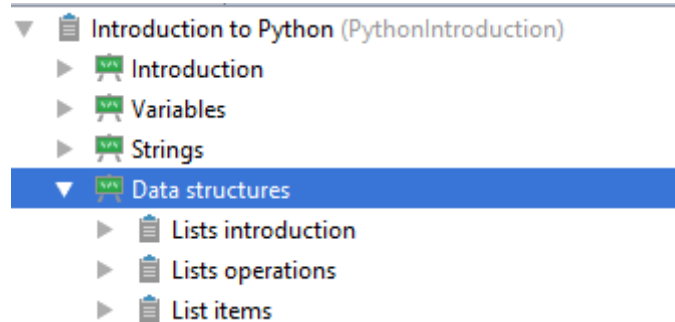


1.3 College 2

1.3.1 Opdracht: Python Introduction

(Her)open het Py-Charm PythonIntroduction project en volg de onderstaande deellessen van de les Data Structures in PyCharm:

- Lists introduction
- Lists operations
- Lists items



1.3.2 Opdracht: Lists

- De trein van Utrecht Centraal naar Maastricht rijdt langs de volgende stations: 's Hertogenbosch, Eindhoven, Weert, Roermond en Sittard.
- 1: zet deze stations in een lijst.
- **2: print het hele traject.**
- 3: de trein wordt herleid van Sittard naar Heerlen. Vervang in de lijst Maastricht door Heerlen.
- 4: Er wordt na Roermond een extra tussenstop gemaakt in Echt. Voeg toe aan de lijst

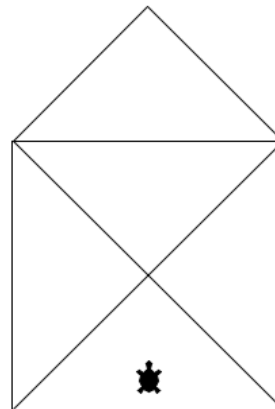
1.3.3 Opdracht: Turtle-huis

Teken het huis zoals hiernaast afgebeeld voor de turtle zonder de functie penup() te gebruiken.

Alle lijnen moeten dus aansluitend worden getekend.

Nadat het huis getekend is, mag je weer penup() gebruiken en plaats je de turtle in het huis.

tip: maak eerst een PSD



1.3.4 Opdracht: Maak een PSD:

Gegeven:

- Roxanne wil haar OV-chipkaart opladen. Ze houdt haar ov-chipkaart voor de kaartscanner en de automaat herkent de ov-chipkaart. Ze kiest voor de optie "opladen" en kiest voor "Betalen met pinpas". Ze voert haar pinpas in en betaalt. Ze houdt haar ov-chipkaart voor de kaartscanner en haar kaart wordt opgeladen.
- Gebruik de tool Structorizer: <http://structorizer.fisch.lu>

1.3.5 Opdracht: Functies

- Maak op basis van je PSD de functies voor de casus "Roxanne laadt haar ov-chipkaart op". Maak voor elke stap een functie die een string print waarin staat aangegeven wat er in die stap gebeurt.

1.4 Weekopdracht

In deze opdracht ga je een nieuwe kaartautomaat voor de NS maken. De volgende onderwerpen komen aan bod:

- lists
- strings
- operators
- van taak naar ontwerp
- functies
- turtle drawing

Gegeven is het volgende treintraject Schagen - Maastricht:

Schagen, Heerhugowaard, Alkmaar, Castricum, Zaandam, Amsterdam Sloterdijk, Amsterdam Centraal, Amsterdam Amstel, Utrecht Centraal, 's-Hertogenbosch, Eindhoven, Weert, Roermond, Sittard, Maastricht.

Je gaat nu een kaartautomaat maken waarbij je het begin- en eindstation vraagt. Vervolgens print je de gegevens over de reis, namelijk:

- de plaats van het beginstation binnen het traject (Alkmaar is het derde station)
- de plaats van het eindstation binnen het traject
- de afstand van de reis (in aantal stations)
- de prijs van de reis.

Het resultaat ziet er als volgt uit:

Geef je beginstation in: `Alkmaar`

Geef je eindstation in: `Weert`

Het beginstation Alkmaar is het 3e station in het traject.

Het eindstation Weert is het 12e station in het traject.

De afstand bedraagt 9 stations.

De prijs van het kaartje is 45 euro.

Niet schrikken, we gaan dit in kleine stapjes opbouwen. Daar gaan we!

1.4.1Opdracht: PSD

Maak een PSD van het hele proces

1.4.2Opdracht: Kaartautomaat

Programmeer de kaartautomaat. Om input te vragen aan de gebruiker, gebruik je de functie `input(<tekst>)`.

Het resultaat sla je op in een variabele. Je mag hier uitgaan van een ritprijs van 5 euro van station naar station.

1.4.3Opdracht: Turtle Graphics

Programmeer de kaartautomaat nogmaals, maar nu maak je het geheel visueel aantrekkelijker, door Turtle graphics te gebruiken. De invoer is hierbij dan gelijk, maar bij de uitvoer maak je gebruik van Turtle graphics.

In Figuur 4: Eerste output zie je het resultaat.



FIGUUR 4: EERSTE OUTPUT

1.4.4 Opdracht: Rapportage

Gegeven is de informatie over 9 verschillende ritten en de bijbehorende afstand in kilometers die door individuele klanten van de NS wordt gereisd met de trein:

11, 1, 3, 6, 4, 6, 8, 3, 5

In deze opdracht maak je een rapport voor de marketingafdeling van de NS. In dit rapport staan een aantal gegevens:

- hoeveel reizigers er hebben gereisd
- hoeveel stations er gemiddeld door reizigers wordt gereisd (gemiddelde reisafstand).
- wat de mediaan van alle reisafstanden is.
- wat de minimale reisafstand is.
- wat de maximale reisafstand is.

Het resultaat ziet er uit zoals in Figuur 5: Tweede output



FIGUUR 5: TWEDE OUTPUT

Voor deze opdracht moet je gebruik maken van de module `statistics` nodig.

Om de module te gebruiken, moet je bovenaan je code het statement `import statistics` invoeren.

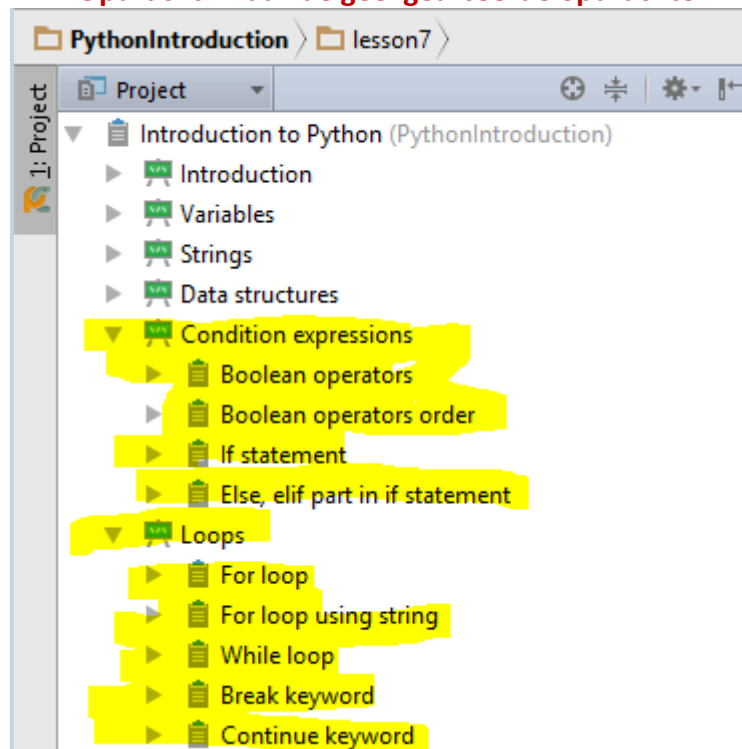
In `statistics` heb je een aantal functies die je kunt gebruiken, zoals het gemiddelde, en de mediaan. Voor meer informatie over `statistics`: <https://docs.python.org/3/library/statistics.html>

2 Week twee: Control flow

2.1 Voorbereiding

We gaan er in deze les vanuit dat je al een beetje weet wat “if”-statements en “for”, “while”-loops zijn. Maak daarom de onderstaande opdrachten voordat je naar het college gaat: Start Pycharm en (re)open het project “PythonIntroduction”.

2.1.1 Opdracht: Maak de geel gearceerde opdrachten:



Het is belangrijk dat je al het geleerde elke week weer even herhaald. Daarom wat extra oefeningen als herhaling en ter voorbereiding op deze week:

2.1.2 Opdracht: Subscripting en Slicing

Gebruik de onderstaande variabele “s” om losse karakters (dat noemen we “subscripting”) en stukjes van de string te tonen (dat noemen we “slicing”).

```
s = "Guido van Rossum heeft de programmeertaal Python bedacht."
```

- print het eerste karakter.
- Print het 11^e karakter.
- print het laatste karakter.
- print de eerste 5 karakters.
- print de laatste 5 karakters.
- print alles vanaf karakter 11.

2.1.3 Opdracht: Invoer en uitvoer

Vraag de gebruiker om zijn/haar naam met de functie “input” en beeld daarna af op het scherm “Welkom <naam> !”

2.1.4 Opdracht: Als/Dan

Vraag de gebruiker “Henk” om zijn wachtwoord met de functie “input” en print “Welkom” als het wachtwoord gelijk is aan “password123”.

2.1.5 Opdracht: For loop

In de Python tutorial “for loop using string” heb je gezien hoe je met een loop langs iedere letter van een string kunt gaan. Gebruik de onderstaande string en een for-loop en print alleen de letter op het scherm als deze letter gelijk is aan de letter “e”.

```
s = "Guido van Rossum heeft de programmeertaal Python bedacht."
```

2.1.6 Opdracht: Sommetjes

Vraag de gebruiker om twee getallen in te voeren en print het kwadraat van deze getallen.

Tip: maak van het ingevoerde getal eerst een integer want “strings” kan je niet met elkaar vermenigvuldigen en integers wel. Voorbeeld om hierbij te helpen:

```
a="3"
print("a is een:", type(a))
b=int(a)
print("b is een:", type(b))
```

2.1.7 Opdracht: Sommetjes 2

Vraag de gebruiker om twee getallen in te voeren en print de som en het verschil van deze getallen.

2.1.8 Opdracht: Sommetjes 3

Vraag de gebruiker om twee getallen in te voeren en print hoe vaak het tweede getal past in het eerste en ook hoeveel je dan nog overhoudt. Gebruik de operators % en //.

2.1.9 Opdracht: Lists

Maak een lijst bestaande uit zes verschillende automerken.

- Print de hele lijst.
- print de eerste auto uit de lijst.
- print de derde auto uit de lijst.
- print de laatste auto uit de lijst.

2.1.10 Opdracht: Functies

Maak een functie met de naam “verdubbel”.

Geef een cijfer mee aan de functie en zorg ervoor dat deze functie het dubbele van de invoer print op het scherm. Zorg ervoor dat deze functie twee keer wordt aangeroepen in je programma. Een keer met als invoer het getal “7” en eenmaal met als invoer het getal “666”.

2.2 College 1

2.2.1 Opdracht: Volle trein, als-dan...

Werk het volgende voorbeeld uit (PSD en code):

De sprinter van Utrecht Centraal (type SLT-IV) naar 's-Hertogenbosch heeft 216 zitplaatsen

- Geef een melding als de trein vol is.
- Tip: Je hebt operators nodig en een “als...dan” constructie. Bekijk eventueel de handouts van de les.

2.2.2 Opdracht: Volle trein, als-dan...anders...

Werk het volgende voorbeeld uit (PSD en code):

De sprinter van Utrecht Centraal (type SLT-IV) naar 's-Hertogenbosch heeft 216 zitplaatsen

- Geef een melding als de trein vol is.
- Anders, geef dan de melding “trein is niet vol”.

2.2.3 Opdracht: Volle trein, als-dan...anders-dan... anders...

Werk het volgende voorbeeld uit (PSD en code):

De sprinter van Utrecht Centraal (type SLT-IV) naar 's-Hertogenbosch heeft 216 zitplaatsen

- Geef een melding als de trein vol is.
- Anders, als de trein leeg is, geef dan aan dat de trein leeg is.
- Anders geef je aan dat de trein niet vol is.

2.2.4 Opdracht: Iteratie, Kosten per jaar

Werk het volgende voorbeeld uit (PSD en code):

Roxanne reist iedere dag op en neer tussen Utrecht Centraal en 's-Hertogenbosch

Op en neer reizen kost EUR17

Iedere tiende reis reist ze gratis

- Reken uit hoeveel Roxanne per jaar kwijt is aan treinreizen.
- Uitgaande van 365 dagen per jaar moet je output gelijk zijn aan: “Roxanne is 5593 Euro per jaar kwijt aan treinreizen.”
- Tip: begin met een “for”-loop en gebruik ook een if statement...

2.2.5 Opdracht: Iteratie, “While” versus “For”

Herschrijf de onderstaande code, maar gebruik nu geen “while”, maar in plaats daarvan een “for” loop. Zorg ervoor dat de code dezelfde uitvoer heeft...

```
lijst = [4, 3, 6, 2, 1]
i = 0
while i < len(lijs):
    print(lijs[i])
    i += 1
```

2.2.6 Opdracht: Iteratie, “While” versus “For”

Werk het volgende voorbeeld uit (PSD en code):

Gegeven is het volgende treintraject:

Woerden

Vleuten

Utrecht Terwijde

Utrecht Leidsche Rijn

Utrecht Centraal

Zolang we nog niet bij het eindstation zijn, print het huidige station

Gebruik een while-loop, maar plaats de stationsnamen eerst in een lijst.

2.3 College 2

2.3.1 Opdracht: FC Utrecht fans

Neem het volgende traject: *Amsterdam Centraal, Amsterdam Amstel, Utrecht Centraal, 's-Hertogenbosch*

- Print ieder station
- Maar: stop (met printen) bij Utrecht, want daar moeten de supporters uitstappen
- print dan “eindstation F.C.Utrecht”
- Gebruik hierbij “break” en een loop (“while” of “for”)

2.3.2 Opdracht: Ajax fans

Neem het volgende traject: *'s-Hertogenbosch, Utrecht Centraal, Amsterdam Amstel, Amsterdam Centraal*

- Print ieder station
- Maar: Sla station Utrecht Centraal altijd over, want daar willen de Ajax-fans niet stoppen
- Gebruik hierbij “continue”

2.4 Weekopdracht

In deze opdracht ga je “if-elif-else” en “for” / “while” loops toepassen in een grotere opdracht.

2.4.1 Opdracht: Plan je route

Gegeven is het volgende treintraject:

Schagen, Heerhugowaard, Alkmaar, Castricum, Zaandam, Amsterdam Sloterdijk, Amsterdam Centraal, Amsterdam Amstel, Utrecht Centraal, 's-Hertogenbosch, Eindhoven, Weert, Roermond, Sittard, Maastricht

Vraag de gebruiker om het beginstation en eindstation, maar bouw nu een check in of de juiste invoer wordt gegeven: een station dat deel uitmaakt van het traject. Blijf dezelfde vraag stellen totdat de gebruiker een station noemt dat wél deel uitmaakt. Er is één vereiste: Je moet een while-loop gebruiken.

De output ziet er als volgt uit:

```
Geef je beginstation in:Alkmaar
Geef je eindstation in:Weert
Geef je eindstation in:Weert
Het beginstation is Alkmaar en het eindstation is Weert
```

- Maak een PSD en programmeer daarna de pythoncode.
- Extra opdracht: gebruik hiervoor dezelfde Turtle-graphics (met input-venster) die je in de opdracht van Unit 1 gebruikte.

2.4.2 Opdracht: Route informatie

Gebruik hetzelfde treintraject. Stel je voor dat er een trein is die dit traject rijdt. Zorg dat er tijdens de treinreis bij ieder station wordt omgeroepen welke stations er nog volgen. Voorbeeld: Als je bij station Roermond bent, dan krijg je de melding:

```
Het huidige station is Roermond.
De resterende stations zijn:
Sittard
Maastricht
```

Bij het laatste station krijg je de melding:

```
Huidige station is Maastricht
Dit is het eindpunt.
```

- Zorg dat dit ook voor de andere stations geldt.
- dus bij ieder station in het traject Schagen - Maastricht moet er melding worden gegeven. Tip: je hebt hier een zogeheten nested for-loop nodig: een for-loop in een for-loop. Kijk hier voor meer informatie: http://www.tutorialspoint.com/python/python_nested_loops.htm
- Maak een PSD en programmeer daarna de pythoncode waarbij alle meldingen voor het hele traject worden geprint (met een lege regel als er een nieuw station wordt aangedaan).

2.4.3 Opdracht: Extra beveiliging van de e-ticket

Als extra beveiliging wil de NS op haar E-ticket nog een unieke code afbeelden. Er is gekozen voor een hele eenvoudige beveiliging: Neem de naam van de gebruiker+beginstation+eindstation, vertaal elk karakter naar ascii en tel er 3 waardes bij op. De "a" wordt dus een "d". Zorg ervoor dat er maximaal 20 karakters worden afgebeeld.



FIGUUR 6: <http://www.automatiseringgids.nl/nieuws/2009/13/ns-overvallen-door-succes-e-tickets> | ASCII TABEL

Bijvoorbeeld:

E-ticket: "Jansen Utrecht Amsterdam CS"

Jouw uitvoer: Mdqvhq#XwuhfkW#Dpvh

- Zorg ervoor dat je code maakt die deze uitvoer maakt op basis van de invoer.
- Tips: gebruik de functies ord() en chr()

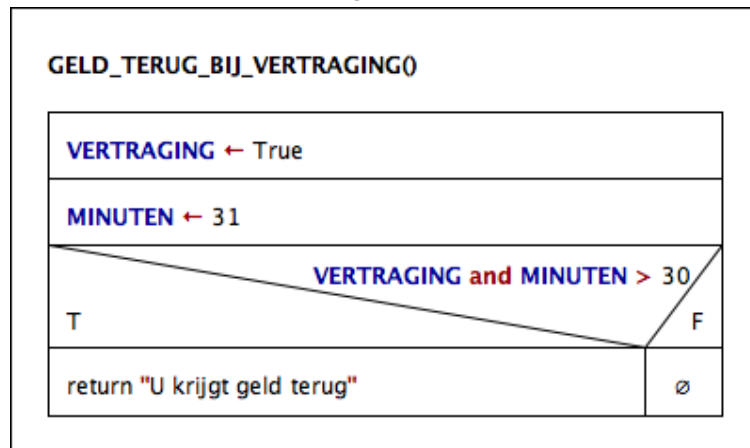
3 Week drie: Expressies en documentatie

3.1 Voorbereiding

3.1.1 Logical Operators: AND | OR | XOR

“Als ik vertraging heb **en** de vertraging bedraagt meer dan 30 minuten, dan krijg ik geld terug”.

In een Nassi-Shneiderman Diagram (NSD) is dit:



Gecodeerd is dit:

```

def geld_terug_bij_vertraging():
    vertraging=True
    minuten=31
    if vertraging and minuten > 30:
        return "U krijgt geld terug"

print(geld_terug_bij_vertraging())
  
```

“Als er blaadjes op het spoor liggen **of** de bovenleiding is kapot, dan rijden we een aangepaste dienstregeling”.

Gecodeerd is dit:

```

def aangepaste_dienstregeling():
    blaadjes = True
    bovenleiding_kapot = True
    if blaadjes or bovenleiding_kapot:
        return "We rijden een aangepaste dienstregeling"

print(aangepaste_dienstregeling())
  
```

Naast “AND” en “OR” bestaat er nog een logische operator in Python: XOR

Als... XOR... dan...

XOR = exclusive OR

Slechts één van beide condities mag waar zijn:

- Letter N in de naam van een reiziger
- Letter S in de naam van een reiziger

string	N	S	XOR-waarde
Rosanne	<u>False</u>	<u>False</u>	<u>False</u>
Nick	True	<u>False</u>	True
Simon	<u>False</u>	True	True
Nick en Simon	True	True	<u>False</u>

let op, Python vergelijkt hoofdlettergevoelig en daarom is er geen match bij de 's' en 'n' ins de naam 'Rosanne'.

Gecodeerd is dit:

```
def een_van_de_twee(naam):
    if (naam.count("N") > 0) ^ (naam.count("S") > 0):
        return naam + ": XOR-test gelukt."
    else:
        return naam + ": XOR-test niet gelukt"

print(een_van_de_twee("Rosanne"))      # XOR-test niet gelukt
print(een_van_de_twee("Nick"))         # XOR-test gelukt.
print(een_van_de_twee("Simon"))        # XOR-test gelukt.
print(een_van_de_twee("Nick en Simon")) # XOR-test niet gelukt
```

3.1.2Opdracht: Logical Operators

Neem de volgende ov-chipkaartnummers:

124576
795834
890432
907251

De NS heeft een leuke actie. Elke reiziger "met een ov-kaartnummer dat deelbaar is door 3" **XOR** "een ov-kaartnummer dat eindigt op een 4" krijgt korting vandaag. Zet de vier getallen in een lijst en doorloop deze lijst. Print voor iedere chipkaart het nummer en print daarnaast "korting" als deze reiziger recht heeft op korting...

Tip: Om te zien of een getal eindigt op een "4" maak je er eerst een string van en dan vergelijk je het laatste karakter met een "4": `(str(i)[-1] == "4")`

3.1.3 Membership Operators: in | not in

Als je wilt testen of iets voorkomt in een string, list, etc.:

```
def n_en_s(naam):
    if ("N" in naam) and ("S" not in naam):
        return naam + ": Er zit een N maar geen S in de naam."

print(n_en_s("Simon"))
print(n_en_s("Nick"))
print(n_en_s("Nick en Simon"))
```

3.1.4 Opdracht: Logical Operators

Neem de volgende ov-chipkaartnummers:

```
124576
795834
890432
907251
```

De NS heeft alweer een leuke actie. Elke reiziger “met een ov-kaartnummer dat deelbaar is door 3” **EN** “een ov-kaartnummer waar een “4” in voorkomt krijgt korting vandaag. Zet de vier getallen in een lijst en doorloop deze lijst.

Print voor iedere chipkaart het nummer en print daarnaast “kortings” als deze reiziger recht heeft op korting...

Tip: Het is niet altijd nodig, maar vaak wel duidelijk om haakjes te gebruiken om de verschillende beweringen en de operators van elkaar te scheiden. Gebruik in je antwoord dus een vergelijking die lijkt op:

```
if (...) and (...):
```

3.1.5 Opdracht: Logical Operators

Neem de volgende ov-chipkaartnummers:

```
124576
795834
890432
907251
```

De NS heeft een actie. Elke reiziger “met een ov-kaartnummer dat geen 4 bevat” **EN** “een nummer waar wel een 9 in voorkomt” krijgt korting vandaag. Zet de vier getallen in een lijst en doorloop deze lijst.

Print voor iedere chipkaart het nummer en print daarnaast “kortings” als deze reiziger recht heeft op korting...

3.1.6 Identity operators: is | is not

Als je wilt testen of twee variabelen dezelfde identiteit hebben kan je “is” gebruiken:

```
var1 = "Nick and Simon"
var2 = var1
print(var1 is var2) #True
```

Het lijkt erop dat “is” gelijk is aan “==”, maar dat is niet waar:

```
== vergelijkt of de twee waarden gelijk zijn, dus 333==111*3 #True
is vergelijkt of de identiteit van beide objecten gelijk zijn, zie het voorbeeld hierboven.
```

Daarom is het onderstaande “False”, immers het gaat om verschillende objecten:

```
print(333 is 3*111) #False
```

3.1.7 Meer informatie:

- Perkovic Hoofdstuk 2
- logical operators:
 - <http://openbookproject.net/thinkcs/python/english3e/conditionals.html>
 - http://www.programiz.com/python-programming/operators#logical_operators
- membership operators:
 - http://www.tutorialspoint.com/python/membership_operators_example.htm
 - <https://docs.python.org/3/reference/expressions.html#not-in>
- identity operators:
 - http://www.tutorialspoint.com/python/identity_operators_example.htm
 - <http://stackoverflow.com/questions/132988/is-there-a-difference-between-and-is-in-python>

3.1.8 Docstrings:

Je maakt tot nu toe maar korte scripts waarbij vrij duidelijk is waar het over gaat. Als je langere scripts maakt, bijvoorbeeld straks bij een mini-project, dan is het belangrijk dat je jouw code goed documenteert zodat de gebruiker weet hoe het werkt, maar ook zodat jij of een andere developer snapt wat je code doet. Het documenteren van je code kan je doen door af en toe een commentaar regel te gebruiken, dat is een regel voorafgaand met een hekje (#):

```
# Hieronder een lijstje met verschillende ov-card nummers
l=[124576, 795834, 890432, 907251]
```

Het is een goede gewoonte om al je functies, methodes, klassen en modules te laten beginnen met een “docstring”. Een docstring is een commentaarregel beginnend met “""" en eindigende met “""" Dit is de eerste zin onder een stuk code zoals een functie. Je beschrijft er het gedrag van je functie. Bijvoorbeeld:

```
def calculate_square(n):
    """Returns the square of a number"""
    return n * n
```

Het is nog beter om de functie iets uitgebreider te beschrijven. Dit noemen we een “multiline docstring”. Je beschrijft dan ook wat je meegeeft aan parameters, maar ook wat de functie teruggeeft (return). Het bovenstaande voorbeeld wordt dan:

```
def calculate_square (n):
    """Calculate the square of a given number.
    Args:
        n: The number to calculate the square of.
    Return:
        The square of the number.
    """
    return n * n
```

Een gebruiker of een developer kan je commentaar vervolgens natuurlijk lezen in je script, maar je kan het ook opvragen. Voor het bovenstaande voorbeeld kan dit bijvoorbeeld zo:

```
print( help(calculate_square) )
```

3.1.9 Opdracht: Docstrings

Gegeven is de onderstaande code. Type deze over en documenteer de functie “dobbelsteen” vervolgens met behulp van een multiline docstring en gebruik daaronder de help functie om ervoor te zorgen dat je de helptekst uitvoert op het scherm.

```
import random

def dobbelsteen(min=1, max=6):
    x=random.randrange(min,max+1)
    return x

z=dobbelsteen()
print("Random waarde:",z)

z=dobbelsteen(max=10)
print("Random waarde:",z)
```

3.1.10 Meer informatie:

- Docstrings: <http://www.python.org/dev/peps/pep-0257/>

3.2 College 1

3.2.1 Opdracht: Exceptions, Stap 1

Je reist met een grote groep en je hebt hiervoor de hele trein afgehuurd. Dat is natuurlijk nogal kostbaar en daarom wil je weten hoeveel dit gaat kosten per persoon. Het totale bedrag is 4356 euro.

Maak een applicatie die de gebruiker vraag om het aantal personen dat mee gaat op reis. Daarna deel je het totale bedrag door het aantal reizigers en dat is dan ook je uitvoer.

Maak deze applicatie.

3.2.2 Opdracht: Exceptions, Stap 2

Er gaat vast iets fout in je code als een gebruiker het aantal "0" invoert in plaats van een positief getal. immers "delen door nul" kan niet. Maak gebruik van try-except om deze fouten op te vangen.

3.2.3 Opdracht: Exceptions, Stap 3

Er kan nog meer fout gaan: de gebruiker kan ook letters zoals het woord "stop" invoeren. Zorg er nu voor dat:

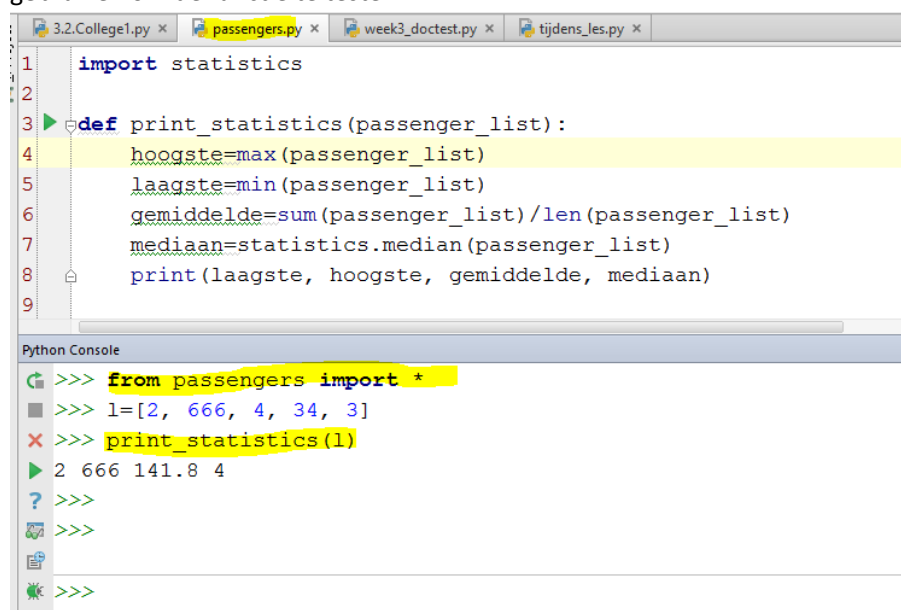
- bij delen door 0 de melding "*delen door nul mag niet*" wordt gegeven en...
- bij alle andere fouten de melding "*onjuiste invoer*" wordt gegeven.

3.2.4 Opdracht: Doctest

De NS wil graag statistische informatie over het aantal passagiers in de trein. Daarom hebben programmeurs een functie geschreven waarin, gegeven het aantal passagiers per dag in de trein, wordt getoond wat het minimum, het maximum, het gemiddelde en de mediaan is van deze aantallen.

Elke dag worden de aantallen passagiers op een traject per trein bijgehouden in een lijst. Bijvoorbeeld, op een bepaald traject reden 5 treinen en daarin zaten in de eerste rit 2, daarna 666, daarna 4, daarna 34 en op de laatste rit maar 3 passagiers. Dat levert de lijst: [2, 666, 4, 34, 3]

Zorg er nu voor met behulp van doctest dat je deze functie goed kan testen. Zie ook de informatie in de slides en de afbeelding hieronder. Zorg voor tenminste 5 testscenario's waarbij je lijsten met steeds andere waarden kunt gebruiken om de functie te testen.



```

1 import statistics
2
3 def print_statistics(passenger_list):
4     hoogste=max(passenger_list)
5     laagste=min(passenger_list)
6     gemiddelde=sum(passenger_list)/len(passenger_list)
7     mediaan=statistics.median(passenger_list)
8     print(laagste, hoogste, gemiddelde, mediaan)
9
Python Console
>>> from passengers import *
>>> l=[2, 666, 4, 34, 3]
>>> print_statistics(l)
2 666 141.8 4
>>>
>>>
>>>

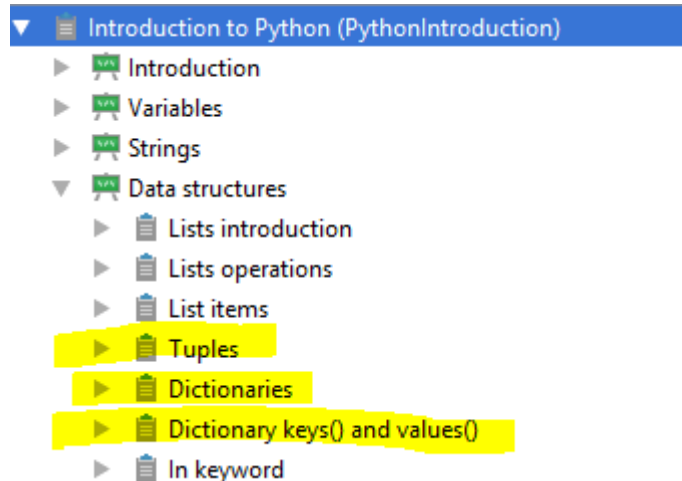
```

3.3 College 2

3.3.1 Opdracht 1: Tuples en Dictionaries

Start Pycharm en (re)open het project “PythonIntroduction”.

Maak daarna de geel gearceerde opdrachten:



3.3.2 Opdracht: Dictionaries en stationsvoorzieningen

Kijk op ns.nl welke voorzieningen station Utrecht heeft, en zet 5 hiervan in een dictionary

- minimaal één voorziening met een string waarde (tekst dus).
- minimaal één voorziening met een boolean waarde (True of False dus).
- minimaal één voorziening met een integer waarde (een geheel getal).

Loop vervolgens door de dictionary en print de voorzieningen met hun bijbehorende waarden op het scherm.

3.3.3 Opdracht: Functie die een tuple teruggeeft.

Zoek de vertrektijden vanaf Utrecht Centraal op voor drie treinen. Je mag gebruik maken van de onderstaande code:

```
spoor=2
traject = {
    'Utrecht': '10:08',
    'Amersfoort': '10:28',
    'Zwolle': '11:25'
}
```

Maak hiervoor een functie met de naam `spoor_en_tijd`

- invoer: string met stationsnaam
- gedrag: kijkt naar de inhoud van de naam van het station en zoekt de juiste waarden op.
- uitvoer: tuple met vertrektijd en spoor (met altijd de waarde “2”)

Roep de functie `spoor_en_tijd` driemaal aan en print de resulterende tuple.

3.3.4 Opdracht: Sets.

- Hieronder staat een afbeelding van een tweetal trajecten (bruin + groen). Zet beide trajecten in een set met de naam “bruin” en de tweede set met de naam “groen”.



FIGUUR 7: GEDEELTE VAN: [HTTP://WWW.ASTRO.RUG.NL/~ISLANDS/SPOORKAART.PDF](http://www.astro.rug.nl/~islands/spoorkaart.pdf)

- Bepaal vervolgens welke plaatsen in beide trajecten worden aangedaan (de overeenkomst tussen de trajecten).
- Bepaal daarna waarin het traject “bruin” verschilt van het traject “groen” (welke plaatsen zijn niet overlappend).

3.3.5Opdracht: Dictionary in dictionary.

Zoek 3 extra voorzieningen van NS-station Utrecht Centraal op, en geef ze weer in een dict-in-dict:

- dict 1: naam van de voorziening
- dict 2: type voorziening, locatie
- values: string-waarde van de type voorziening en de string-waarde van de locatie

Tip: Hieronder vind je de basis met al twee voorzieningen. Maak hier gebruik van en maak de dict-in-dict compleet.

```
voorziening = {
    'NS-service- en verkooppunt': {'type': 'verkoop- en reisinformatie',
                                    'locatie': 'spoor 11/12'},
    'Starbucks': {'type': 'Winkels en restaurants', 'locatie': 'spoor 18/19'}
}
```

- Print daarna met behulp van een for-loop de vijf voorzieningen op de stations, bijvoorbeeld:

```
...
NS-service- en verkooppunt is van het type: verkoop- en reisinformatie en bevindt
zich op spoor 11/12
Starbucks is van het type: Winkels en restaurants en bevindt zich op spoor 18/19
WC is van type: Algemene voorzieningen en bevindt zich op: busstation west
...
```

3.4 Weekopdracht

Deze opdracht betreft een vrije opdracht waar je in de context van de NS een kleine applicatie moet maken. Het is de bedoeling dat je de tot nu toe geleerde stof daarin terug laat komen qua Python-functionaliiteit.

3.4.1 Opdracht: Beschrijving

Bedenk een (kleine) applicatie voor de NS waarin je bijvoorbeeld de conducteur, de reiziger, de machinist, of de railcatering helpt. Je zou kunnen denken aan een kaartautomaat, routeplanner, data-analyse-software die alle reizen in een grafiek/histogram zet (gebruik eventueel matplotlib), een in/uitcheckpoort, of een NS-consumptie-automaat. Je bent vrij om zelf een leuke applicatie te bedenken.

3.4.2 Opdracht: Vereisten

Het is de bedoeling dat de applicatie gebruik maakt van de volgende technieken:

- if-statement (met expressies)
- for- of een while-loop
- list, tuple, set en/of dictionary om gegevens op te slaan
- functies (die hergebruikt kunnen worden)
- input-mogelijkheid (om als gebruiker te kunnen interacteren met de applicatie)



3.5 Extra Uitdagende Opdracht: Spoor netwerk

Studenten die zijn ingedeeld in de groep “extra uitdaging” maken naast de weekopdracht ook de onderstaande opdracht met extra uitdaging.

Neem de stations Duivendrecht, Amsterdam Centraal en Schiphol. Je kunt vanaf Duivendrecht via Amsterdam Zuid naar Schiphol reizen, maar je kunt ook via Amsterdam Centraal en Amsterdam Sloterdijk naar Schiphol reizen.

Sla de twee trajecten Duivendrecht-Schiphol en Duivendrecht-Amsterdam Centraal-Schiphol op door middel van een list-in-list. Dus: modelleer dit spoor netwerk volgens het voorbeeld in de slides van week 3, college 2.

3.5.1 Extra Uitdagende Opdracht: Kortste route (minste stations) van Duivendrecht naar Schiphol

Bereken vervolgens op de kortste route (minste stations) van Duivendrecht naar Schiphol. Je gebruikt hiervoor een while loop, maar je mag een station niet twee keer bezoeken, omdat je dan in een oneindige loop terecht kan komen. Houd hier dus rekening mee.

3.5.2 Extra Uitdagende Opdracht: Voorzieningen per station

Zoek nu voor een aantal stations op wat de voorzieningen zijn. Dit doe je via de volgende URL:

<http://www.ns.nl/reizigers/reisinformatie/stationsvoorzieningen>

Voeg dit toe aan de informatie die je al hebt over de stations.

Print voor de twee trajecten Duivendrecht – Amsterdam Centraal – Schiphol en Duivendrecht - Amsterdam-Zuid – Schiphol een overzicht van de voorzieningen per station.

4 Week vier: Data

4.1 Voorbereiding

4.1.1 Extra uitdagende opdracht:

Je mag opdracht 4.1.2 overslaan want we gaan ter voorbereiding van deze week wat anders doen.

Het is voor het tweede college belangrijk dat je kennis krijgt van relationele databases in het algemeen en SQLite in het bijzonder.

Een goede manier om in korte tijd wat basale kennis over een nieuwe technologie te krijgen is door de tutorials van w3schools. te volgen: <http://www.w3schools.com/sql/>

Zorg ervoor dat je ten minste de volgende hoofdstukken hebt uitgevoerd, dus niet alleen lezen, voordat je naar het tweede college komt:

SQL HOME | SQL Intro | SQL Syntax | SQL Where | SQL And & Or | SQL Order By | SQL Insert Into | SQL Update | SQL Delete | SQL Like | SQL Create Table

Kijk ook naar een aantal SQL functies:

SQL Count() | SQL Max() | SQL Min() | SQL Sum()

4.1.2 Opdracht:

Je kan het beste leren programmeren als je de opgedane kennis vaak herhaalt. Vandaar de volgende opdracht waarmee je de opgedane kennis nog even goed kan oefenen:

- Gegeven is de onderstaande code:

```
import random
def select_winner(persons):
    """Select a random item from a given list"""
    print("De winnaar wordt gekozen uit de volgende lijst:")
    for person in persons:
        print(person, end=' ')
    print()
    return random.choice(persons)
```

- Maak een "list" met daarin 10 namen.
- Vraag de gebruiker om zijn/haar naam in te voeren. Als de gebruiker een lege waarde invoert, dus alleen op "Enter" heeft geklikt, vraag de gebruiker dan opnieuw de naam in te voeren net zolang tot het goed gaat.
- Voeg die naam toe aan de lijst en sorteer de lijst alfabetisch.
- Roep de bovenstaande functie "select_winner" aan.
- Als de winnaar gelijk is aan de naam van de gebruiker die net de naam heeft ingevoerd print dan: "U heeft een gratis ov-kaart gewonnen". In alle andere gevallen print je "Helaas, u heeft niet gewonnen. De winnaar is: <naam van de winnaar>".

4.2 College 1

4.2.1 Opdracht: Een interactieve gebruiker

Vraag de gebruiker om zijn / haar leeftijd. Print “pensioengerechtigd” als de leeftijd ouder is dan 67 jaar.

4.2.2 Opdracht: Format je uitvoer

Gegeven de onderstaande twee lijsten. zorg ervoor dat je deze informatie “recht onder elkaar” kunt printen.

Maak hiervoor gebruik van de “.format” functie van Python.

```
naam=[ 'jan', 'achmed', 'alice', 'mohammed', 'truus', 'eva', 'matthijs',  
       'marleen' ]  
salaris=[ 900, 1300, 3000, 4100, 3400, 5000, 2800, 10000 ]
```

Uitvoer is dus:

```
Naam: jan          Salaris:          900  
Naam: achmed       Salaris:          1300  
enz...
```

4.2.3 Opdracht: Lezen van files

Maak met behulp van notepad het onderstaande bestand bestaande uit chipkaartnummers en namen:

```
325255, Jan Jansen  
334343, Erik Materus  
235434, Ali Ahson  
645345, Eva Versteeg  
534545, Jan de Wilde  
345355, Henk de Vries
```

Open dit bestand met Python.

Splits elke regel op de komma en zorg ervoor dat de uitvoer als volgt wordt weergegeven:

```
OV-nummer: 325255 is van Jan Jansen  
OV-nummer: 334343 is van Erik Materus  
OV-nummer: 235434 is van Ali Ahson  
OV-nummer: 645345 is van Eva Versteeg  
OV-nummer: 534545 is van Jan de Wilde  
OV-nummer: 345355 is van Henk de Vries
```

Vergeet het bestand niet te sluiten!

4.2.4 Opdracht: Schrijven van files

De NS heeft een wedstrijd: Elke deelnemer reist op eigen wijze van Utrecht naar Groningen. Zodra Groningen is bereikt staat er een computer waar de gebruiker zijn/haar naam invoert. Zodra dit is gebeurd schrijft de computer de naam van de gebruiker + de huidige datum/tijd in een bestand met de naam “resultaat.txt”. Elke gebruiker op een nieuwe regel. Maak de applicatie waarbij je gebruik maakt van de onderstaande code:

```
import datetime  
today = datetime.datetime.today()  
s = today.strftime("%a %d %b %Y, %H:%M:%S")  
print (s)
```

4.3 College 2

4.3.1 Opdracht: Lezen van CSV-bestanden.

Maak met behulp van MS Excel een werkblad aan en sla deze op als csv-file.

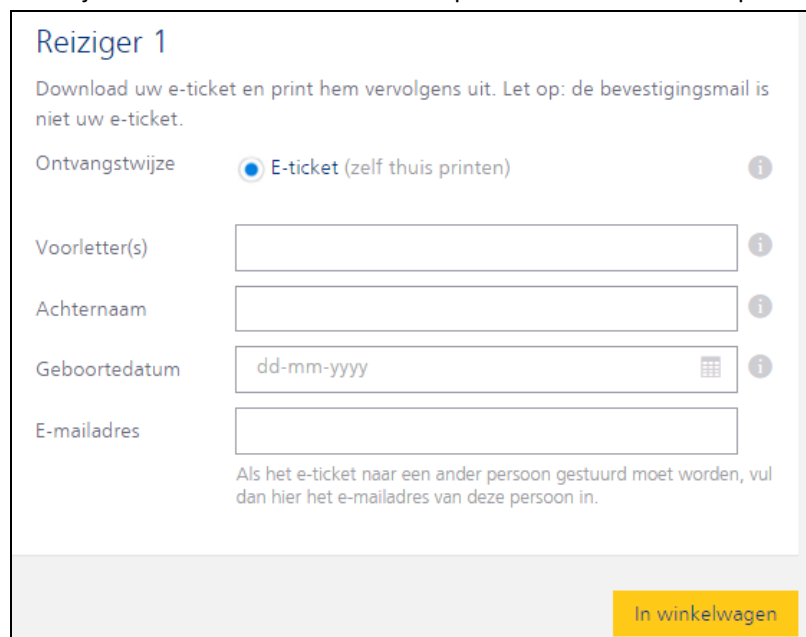
#	Baanvak	Snelheid (km/u)
1	Oss - 's Hertogenbosch	140
2	Roermond - Maastricht	130
3	Hoofddorp - Den Haag Mariahoeve	160
4	Amersfoort - Zwolle	140
5	Woerden - Alphen a/d Rijn	140

Lees de informatie met de Python csv-module en print alleen de baanvakken en snelheden waarbij 140 km/u mag worden gereden.

O ja, zorg ervoor dat de output (baanvak / snelheden) recht onder elkaar komt te staan...

4.3.2 Opdracht: Schrijven van CSV-bestanden.

Wie bij de NS online een e-ticket wil kopen die moet een aantal persoonlijke gegevens opvoeren:



FIGUUR 8: BRON [HTTPS://WWW.NS.NL/PRODUCTEN/S/RETOUR](https://www.ns.nl/producten/s/retour)

Zorg ervoor dat deze gegevens niet verloren gaan, maar worden opgeslagen in een CSV bestand. Kortom, heel concreet:

Vraag de gebruiker om de gegevens en schrijf deze informatie in een CSV file met 4 kolommen voor het opslaan van de NAW gegevens, dwz: Voorletters, Achternaam, Geboortedatum en E-mail adres.

4.4 Weekopdracht

Een station bevat allerlei voorzieningen. Eén ervan is de “bagagekuis”, zie afbeelding. We willen graag dat jij de software realiseert voor deze kuis. De gebruiker ziet het onderstaande menu:

- 1: Ik wil een nieuwe kuis
- 2: Ik wil mijn kuis openen
- 3: Ik geef mijn kuis terug
- 4: Ik wil weten hoeveel kluizen nog vrij zijn

De beheerders hebben de volgende informatie/eisen met betrekking tot het systeem:

- Er zijn 12 kluizen
- De kuis moet beveiligd zijn met een wachtwoord van 4 cijfers.
- Als de stroom uitvalt, mag de data niet verloren gaan (daarom moet je een CSV-file gebruiken voor opslag van data zoals kuisnummers en wachtwoorden).

4.4.1 Extra Uitdagende Opdracht.

De studenten die gaan voor een “Extra uitdagende opdracht” voeren deze weekopdracht uit, alleen maken jullie geen gebruik maken van een CSV-bestand, maar van een database. Jullie implementeren alle onderstaande functionaliteit maar overal waar staat “CSV” gaan jullie gebruik maken van een database.

Jullie implementeren ook optie 3 “ik geef mijn kuis terug”, want dat is met een “DELETE” commando eenvoudig mogelijk in een database.

Hieronder wat extra informatie. Het komt niet precies overeen met jullie opdracht, maar je kan er je informatie vinden die je nodig hebt om een database te maken met “Create table”, te vullen met “Insert” en uit te lezen met “Select”...

Tabel: dienstregeling

id	tijd	perron	type	bestemming
1	08:05	2	stoptrein	Amsterdam
2	09:05	2	stoptrein	Amsterdam
3	10:05	3	sneltrein	Gouda

```
SELECT tijd, perron
FROM dienstregeling
WHERE id=1
```

Veel software maakt gebruik van **relationele databases** voor het opslaan van informatie. Deze informatie wordt vastgelegd in **tabellen**.

Het opvragen van deze informatie gaat via **SQL** (Structured Query Language)

Python: CREATE / INSERT

```
conn=sqlite3.connect(db)
c=conn.cursor()

c.execute("Create table users ('id' integer,
'username' text, 'password' text)")

c.execute("insert into users values(1, 'jan',
'pa$$word')")
c.execute("insert into users values(2, 'els',
'admin123')")

conn.commit()
```

Python: SELECT a single row



```
c.execute("insert into users values(1, 'jan',
'pa$word')")
c.execute("insert into users values(2, 'els',
'admin123')")

c.execute('SELECT * from users')

print("EERSTE USER:", c.fetchone())
print("VOLGENDE USER:", c.fetchone())
```

Python: SELECT in een script



```
u = input('Wat is je gebruikersnaam')
p = input('Wat is je wachtwoord')
t=(u,p)

r=c.execute('SELECT * FROM USERS WHERE username=?
and password=?' , t)

for i in r:
    print(i)

conn.close()
```

Meer informatie over SQLite? Perkovic, hoofdstuk 12.1 en 12.2

<https://www.sqlite.org/docs.html>

<http://pymotw.com/2/sqlite3/index.html#module-sqlite3>

<https://docs.python.org/2/library/sqlite3.html>

4.4.2 Opdracht: Maak de onderstaande functionaliteit:

- Als je kiest voor “1: Ik wil een nieuwe kluis”, dan krijgt de gebruiker een willekeurige code van 4 cijfers als er nog een kluis beschikbaar is. Deze code hoort bij een van de twaalf kluisen. Sla het nummer van de kluis samen met de 4-cijferige code op in een CSV-file en geef de code en het kluisnummer terug aan de gebruiker op het scherm.
- Als je kiest voor “2: Ik wil mijn kluis openen”, dan vraag je de gebruiker om zijn/haar code en als deze klopt dan toon je het nummer van de kluis dat daarbij hoort. (Het zou leuk zijn als we de kluis ook echt konden openen, maar dat gaat lastig zonder een echte set bagagekluisen natuurlijk).
- Als je kiest voor “3: Ik geef mijn kluis terug”, dan zou je eigenlijk een regel uit het csv bestand moeten halen, maar dat is wat ingewikkeld en daarom hoeft deze optie niet in te bouwen.
- Als je kiest voor “4: Ik wil weten hoeveel kluisen nog vrij zijn”, dan geef je als uitvoer het aantal kluisen van de 12 dat nog beschikbaar is.

Bagagekluizen

Op het station kunt u een bagagekluis huren. Makkelijk in gebruik en elektronisch beveiligd.

Bagagekluis/-depot



FIGUUR 9: BRON

[HTTP://WWW.NS.NL/REIZIGERS/REISINFORMATIE/STATIONSVOORZIENINGEN/VOORZIENINGEN/ALGEMEEN/BAGAGEKLUIZEN](http://www.ns.nl/reizigers/reisinformatie/stationsvoorzieningen/voorzieningen/algemeen/bagagekluizen)

4.4.3 Hoe moet ik beginnen?

Geen idee hoe je dit probleem moet aanpakken? Misschien kan het onderstaande stappenplan je er een beetje bij helpen:

- 1) Maak het menu met de opties 1-4 en zorg dat dit werkt. Maak voor iedere menu optie een aparte functie.
- 2) Zoek een stukje code op het internet of bestudeer de module "random" om een willekeurig getal van 4 cijfers (dat is een getal tussen 1000-9999) te genereren.
- 3) Maak menu optie 1:
 - a. Controleer eerst of je wel een vrije kluis hebt (niet meer dan 12 in gebruik) en zo ja, welke kluis met welk nummer vrij is.
 - b. Maak het willekeurige getal.
 - c. Geef het getal aan de gebruiker.
 - d. Sla kluisnummer en willekeurige getal op in de CSV-file.
- 4) Maak menu optie 2
- 5) Maak menu optie 3 (Een lege functie met "pass" is voldoende).
- 6) Maak menu optie 4 (Tel het aantal regels in de csv file en als dit ≥ 12 is dan meldt je dit aan de gebruiker).

5.2 College 1

5.2.1 Opdracht (10 min): API key aanvragen

- Ga naar <http://www.ns.nl/api/home>
- Vraag toegang aan
- Klik op de link in de email om registratie compleet te maken
- Check je email voor je gebruikersnaam en wachtwoord

5.2.2 Opdracht: XML

Schrijf met notepad, of nog beter en gemakkelijker met Pycharm, een kort xml-document met als root element “perrons”.

Vul deze met 5 regels met telkens het element “perron”. Geef elk perron een unieke naam.

Installeer de module xmltodict en probeer het derde perron af te beelden op het scherm. Dus stel het derde perron heet “zuidzijde” dan moet onderstaande code als uitvoer “zuidzijde” opleveren.

```
def verwerk_xml():
    bestand = open('perrons.xml', 'r')
    xml_string = bestand.read()
    return xmltodict.parse(xml_string)

stations_dict = verwerk_xml()
print(stations_dict['perrons']['perron'][2])
```

5.3 College 2

5.3.1 Opdracht: tkinter

Maak een Window met daarop twee buttons naast elkaar;

Als je op de linker button klikt toon je een pop-up met daarin de tekst: "Links!"

Als je op de rechter button klikt toon je een pop-up met daarin de tekst: "Rechts!"

Gebruik hierbij de voorbeelden uit de presentaties en de informatie hieronder:

```
button.pack (padx=5, pady=10, side=LEFT)
```

Hierin is:

padx = positie op de x-as

pady = positie op de y-as

side = LEFT of RIGHT

Je hoeft de bovenstaande keyword arguments niet allemaal tegelijk te gebruiken overigens.

5.4 Weekopdracht

5.4.1 Casus: Actuele vertrektijden

Het valt het docentencorps op de verschillende scholen in Utrecht op dat studenten altijd snel weg willen uit de les om de trein te halen, maar als hen wordt gevraagd wanneer die trein dan precies vertrekt en wanneer eventueel de volgende trein komt dan lijken studenten dit eigenlijk niet precies weten...

De directie wil daarom graag dat er een computerprogramma wordt gemaakt die daarbij kan helpen: *“Laat alle actuele vertrektijden zien van de treinen naar een bepaalde eindbestemming”* is de opdracht. Zo ben je snel op de hoogte van de vertrektijden van jouw trein!

Daar gaan we, alles in kleine stapjes gebruikmakend van de NS API en wat je hebt geleerd tijdens de lessen over XML en Tkinter...

5.4.2 Opdracht: De NS API

Zorg er eerst voor dat je weet wanneer de treinen vertrekken naar een bepaalde eindbestemming. Dat gaan we in een volgende stap inbouwen in je programma.

Stap 1: Zorg voor een NS key (heb je als het goed is al gedaan in 5.2.1)

Stap 2: Doe vanuit je browser deze vraag: <http://webservices.ns.nl/ns-api-avt?station=ut> Kan je bijvoorbeeld vinden wanneer de eerste trein vertrekt naar Nijmegen? Hoe laat is dat?

Stap 3: Kijk naar de slides van het eerste werkcollege en zorg ervoor dat je met python hetzelfde kan als wat je net deed in de browser. Gebruik dezelfde namen voor de variabelen als in de slides. Print het resultaat (een XML-document) op het scherm.

Meer informatie over deze API is opgenomen in de appendix van dit document. De informatie is ook te lezen op de NS website (www.ns.nl/api).

5.4.3 Opdracht: Loop door de treinen en print steeds een enkele trein.

Je krijgt veel te veel informatie in je XML-file. Die moeten we filteren. We willen telkens maar een enkel trein. Hoe halen we deze informatie nu uit deze grote XML?

Hiervoor gebruiken we de module “xmldict” en maken een loop.

Verwerk onderstaande code in programma:

```
import xmldict
stations_dict = xmldict.parse(response.text)
for i in stations_dict['ActueleVertrekTijden']['VertrekkendeTrein']:
    vertrekkende_trein=dict(i)
    print(vertrekkende_trein)
```

Uitvoer is dan iets als dit:

```
{'RitNummer': '1726', 'RouteTekst': 'Gouda', 'Vervoerder': 'NS', 'EindBestemming': 'Den Haag Centraal', 'TreinSoort': 'Intercity', 'VertrekSpoor': OrderedDict([('wijziging', 'true'), ('#text', '8a')]), 'VertrekTijd': '2015-08-24T09:29:00+0200'}
{'RitNummer': '28301', 'VertrekTijd': '2015-08-24T09:30:00+0200', 'Vervoerder': 'NS', 'EindBestemming': 'Utrecht Maliebaan', 'TreinSoort': 'Sprinter', 'VertrekSpoor': OrderedDict([('wijziging', 'false'), ('#text', '4b')])}
...
```

Je hoeft dit niet helemaal te begrijpen zolang je snap dat je door de XML heen loopt en telkens in de for-loop een enkele trein print. Deze enkele trein “vertrekkende_trein” is nu een dictionary geworden en daaruit kan je de EindBestemming en de VertrekTijd halen.

Werkt dit?

5.4.4 Opdracht: Alleen de vertrektijden van de trein naar “Nijmegen”

Zorg er nu voor dat je alleen `print(vertrekkende_trein)` uitvoert als de eindbestemming van de trein “Nijmegen” is. (dat kan met een `if` – statement).

5.4.5 Opdracht: Andere Eindbestemming.

Laat nu de gebruiker de eindbestemming invoeren. Gebruik hiervoor de functie “input”.

5.4.6 Opdracht: Betere uitvoer.

Print nu het woord “VertrekTijd” met daarachter alleen het uur, minuten en seconden van de vertrektijd. De overige uitvoer mag je weglaten. Dus iets als:

VertrekTijd: 09:29:00

VertrekTijd: 09:30:00

Dat kan je eenvoudig doen door uit de vertrektijd alleen een paar karakters te laten zien van deze string. String “slicing” noemen we dat in Python.

5.4.7 Opdracht: Tkinter

Het programma is klaar, maar de user interface is dat nog niet.

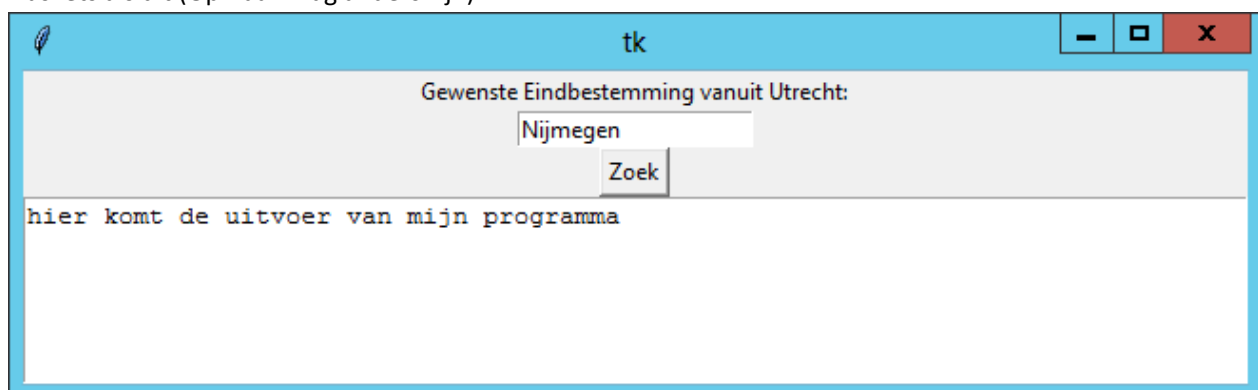
Maak nu een heel nieuw python programma (nieuwe python file).

We gaan met Tkinter een mooie(re) interface maken voor de eindgebruiker.

Wat hebben we nodig:

- Een invoerveld voor de keuze van de eindbestemming.
- Een “zoek” knop die je indrukt zodra de eindbestemming is ingegeven.
- Een uitvoerveld waarin de vertrektijden worden getoond.

Dus iets als dit (Opmaak mag anders zijn):



Zodra de Button “Zoek” wordt ingedrukt print je “hier komt de uitvoer van mijn programma”.

Tip1: De meeste informatie is besproken tijdens het werkcollege. Voor de uitvoer is ook de widget “Tekst” naast de “Button”, de “Entry” en het “Label” nodig. Zoek zelf uit hoe je dit kan gebruiken.

Tip2: met de methode “insert” kan je informatie toevoegen aan de tekst widget. Met de methode “delete” kan je de tekst widget ook weer wissen.

5.4.8 Opdracht: Integratie

Gelukt? Maak nu een derde Python file waarin je beide programma’s integreert.

Het komt er eigenlijk op neer dat je het eerste programma waarin je de vertrektijden opvraag opneemt in de functie die wordt aangeroepen zodra je op de “Zoek” knop klikt.

5.5 Extra Uitdagende Opdracht: Giphy API

In deze opdracht leer je werken met de Giphy API waarmee je GIFs kunt zoeken en weergeven. Bezoek Giphy.com om te kijken hoe het zoekproces werkt. Kijk vervolgens bij de API hoe je verbinding maakt met de server van Giphy: <https://github.com/Giphy/GiphyAPI>.

Hoe de zoekmachine API van Giphy werkt, kun je lezen op <https://github.com/Giphy/GiphyAPI#search-endpoint>

5.5.1 Opdracht: Maak een TKinter scherm waarin je afbeeldingen kunt zoeken en weergeven

Maak een TKinter scherm waarin je:

1. een zoekterm kunt ingeven
2. vervolgens op submit drukt
3. de applicatie verbinding laat maken met de Giphy API om te zoeken
4. via het TKinter scherm een lijst met afbeeldings-URLs laat zien, of nog beter: je laat een aantal afbeeldingen direct op het TKinter scherm zien

Benodigdheden

- Module requests om verbinding te maken met de API's
- Module PIL of pillow om afbeeldingen uit bestand weer te geven in TKinter
- Giphy API key: is al gegeven, zie <https://github.com/Giphy/GiphyAPI>

Appendix: Documentatie NS API

Webservice actuele vertrektijden

Overzicht

De webservice voor de actuele vertrektijden maakt het mogelijk voor een station een actueel overzicht op te vragen van alle vertrekkende treinen voor het komende uur.

Request

Actuele vertrekinformatie kan worden opgehaald met behulp van de volgende url:

[http://webservices.ns.nl/ns-api-avt?station=\\${Naam of afkorting Station}](http://webservices.ns.nl/ns-api-avt?station=${Naam of afkorting Station})

Parameter naam	Omschrijving	Verplicht
station	De code (afkorting) of korte naam of middellange naam of volledige naam of synoniem van de stationsnaam	Ja

Er zullen minimaal 10 vertrektijden worden geretourneerd en minimaal de vertrektijden voor het komende uur. Een voorbeeld request voor de actuele vertrektijden van station Utrecht Centraal:

<http://webservices.ns.nl/ns-api-avt?station=ut>

Omdat vanaf Utrecht Centraal meer dan 10 treinen per uur vertrekken, bevat de response alle vertrekinformatie voor het komende uur.

Een voorbeeld request voor de actuele vertrektijden van station Kampen:

<http://webservices.ns.nl/ns-api-avt?station=Kampen>

Omdat vanaf Kampen 2 treinen per uur vertrekken, worden er 10 vertrektijden geretourneerd.

Response

De response bestaat uit het element ActueleVertrekTijden dat per vertrekkende trein een element VertrekkendeTrein bevat.

Een VertrekkendeTrein kan de volgende elementen hebben:

- RitNummer (1): Identificerend nummer van de trein
- VertrekTijd (1): Vertrektijd van de trein in minuten
- VertrekVertraging (0..1): Vertraging in minuten
- VertrekVertragingTekst (0..1): Vertraging presentatietekst
- EindBestemming (1): Geplande eindbestemming van de trein
- TreinSoort (1): Treinsoort (Intercity, Sprinter etc.)
- RouteTekst (0..1): Verkorte weergave van de route die de trein rijdt. Er worden maximaal 4 stopstations opgenomen, waardoor het meestal een samenvatting is van de volledige route.
- Vervoerder (1): Vervoerder van de trein.

- VertrekSpoor (1): Vertrekspoor van de trein. Het attribuut "wijziging" geeft aan of er wel of niet sprake is van een spoorwijziging.
- ReisTip (0..1): Tekstregel die een tip geeft over de reis. Hiermee wordt de aandacht gevestigd op mogelijk onverwachte kenmerken. Bv. een Intercity die stopt op alle tussengelegen stations.
- Opmerkingen (0..1). Tekstuele opmerkingen bij de vertrekkende trein. Bestaat uit 0 of meer Opmerking-elementen, die de afzonderlijke tekstregels bevatten. Bevat vrijwel altijd actuele ritwijzigingen, zoals "Rijdt vandaag niet".

Voorbeeld

Zie het volgende voorbeeld van een (deel van een) response:

```
<ActueleVertrekTijden>
  <VertrekkendeTrein>
    <RitNummer>7478</RitNummer>
    <VertrekTijd>2012-02-19T22:47:00+0100</VertrekTijd>
    <VertrekVertraging>PT2M</VertrekVertraging>
    <VertrekVertragingTekst>+2 min</VertrekVertragingTekst>
    <EindBestemming>Breukelen</EindBestemming>
    <TreinSoort>Sprinter</TreinSoort>
    <Vervoerder>NS</Vervoerder>
    <VertrekSpoor wijziging="true">7</VertrekSpoor>
  </VertrekkendeTrein>
  <VertrekkendeTrein>
    <RitNummer>20578</RitNummer>
    <VertrekTijd>2012-02-19T22:47:00+0100</VertrekTijd>
    <EindBestemming>Rotterdam Centraal</EindBestemming>
    <TreinSoort>Intercity</TreinSoort>
    <RouteTekst>Gouda, R'dam Alexander</RouteTekst>
    <Vervoerder>NS</Vervoerder>
    <VertrekSpoor wijziging="false">9b</VertrekSpoor>
  </VertrekkendeTrein>
  <VertrekkendeTrein>
    <RitNummer>5683</RitNummer>
    <VertrekTijd>2012-02-19T22:50:00+0100</VertrekTijd>
    <EindBestemming>Zwolle</EindBestemming>
    <TreinSoort>Sprinter</TreinSoort>
    <RouteTekst>Utrecht Overv, Den Dolder, Amersfoort</RouteTekst>
    <Vervoerder>NS</Vervoerder>
    <VertrekSpoor wijziging="false">3</VertrekSpoor>
  </VertrekkendeTrein>
```

Voorbeeld ReisTip

Onderstaand voorbeeld toont hoe een reistip bij een vertrekkende trein is opgenomen.

```
<VertrekkendeTrein>
  <RitNummer>31156</RitNummer>
  <VertrekTijd>2012-02-22T15:32:00+0100</VertrekTijd>
  <EindBestemming>Tiel</EindBestemming>
  <TreinSoort>Stoptrein</TreinSoort>
  <RouteTekst>Elst</RouteTekst>
  <Vervoerder>Syntus</Vervoerder>
  <VertrekSpoor wijziging="false">4</VertrekSpoor>
  <ReisTip>Stopt niet in Arnhem Zuid</ReisTip>
</VertrekkendeTrein>
```

Voorbeelden Opmerkingen

Onderstaand voorbeeld toont hoe een opmerking is opgenomen bij een vertrekkende trein om aan te geven dat deze niet rijdt.

```
<VertrekkendeTrein>
  <RitNummer>669</RitNummer>
  <VertrekTijd>2012-02-22T17:13:00+0100</VertrekTijd>
  <EindBestemming>Rotterdam Centraal</EindBestemming>
  <TreinSoort>Intercity</TreinSoort>
  <RouteTekst>Schiphol, Den Haag HS</RouteTekst>
  <Vervoerder>NS</Vervoerder>
  <VertrekSpoor wijziging="false">15a</VertrekSpoor>
  <Opmerkingen>
    <Opmerking>Rijdt vandaag niet</Opmerking>
  </Opmerkingen>
</VertrekkendeTrein>
```

Onderstaand voorbeeld toont hoe een opmerking is opgenomen bij een vertrekkende trein om aan te geven dat instappen niet is toegestaan. Inhoudelijke toelichting: de Thalys stopt op Schiphol t.b.v. het laten uitstappen van reizigers, het is niet de bedoeling dat reizigers hier instappen.

```
<VertrekkendeTrein>
  <RitNummer>9339</RitNummer>
  <VertrekTijd>2012-02-21T15:30:00+0100</VertrekTijd>
  <VertrekVertraging>PT1M</VertrekVertraging>
  <VertrekVertragingTekst>+1 min</VertrekVertragingTekst>
  <EindBestemming>Amsterdam Centraal</EindBestemming>
  <TreinSoort>Thalys</TreinSoort>
  <Vervoerder>NS Hispeed</Vervoerder>
  <VertrekSpoor wijziging="false">1-2</VertrekSpoor>
  <Opmerkingen>
    <Opmerking>Niet instappen</Opmerking>
  </Opmerkingen>
</VertrekkendeTrein>
```


6 Week zes: Object/classes & source control

6.1 Voorbereiding

6.1.1 "Object/Classes"

Bestuderen ter voorbereiding op het werkcollege:

- Perkovic: Chapter 8 OF NOG BETER:
- Swaroop (Byte of Python): Chapter 12 (zie SharePoint).

6.1.2 "Version Control System: Git en Github"

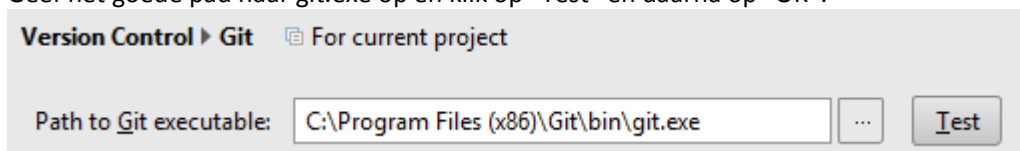
Je hebt de source code geschreven. Natuurlijk wil je niet dat deze source verloren kan gaan of dat je door een vergissing niet meer terug kan naar een eerdere versie van de source. Dit kan je bereiken met een versiebeheersysteem zoals Subversion, TFS of... Git. Deze laatste gaan we gebruiken in combinatie met PyCharm en GitHub.

Git bevat een lokale database, maar het nadeel daarvan is dat je geen back-up hebt als je computer "crashed" en bovendien kan je niet samenwerken aan dezelfde code. Vandaar dat we gebruik maken van Git in combinatie met GitHub waarmee je de code opslaat "in de cloud".

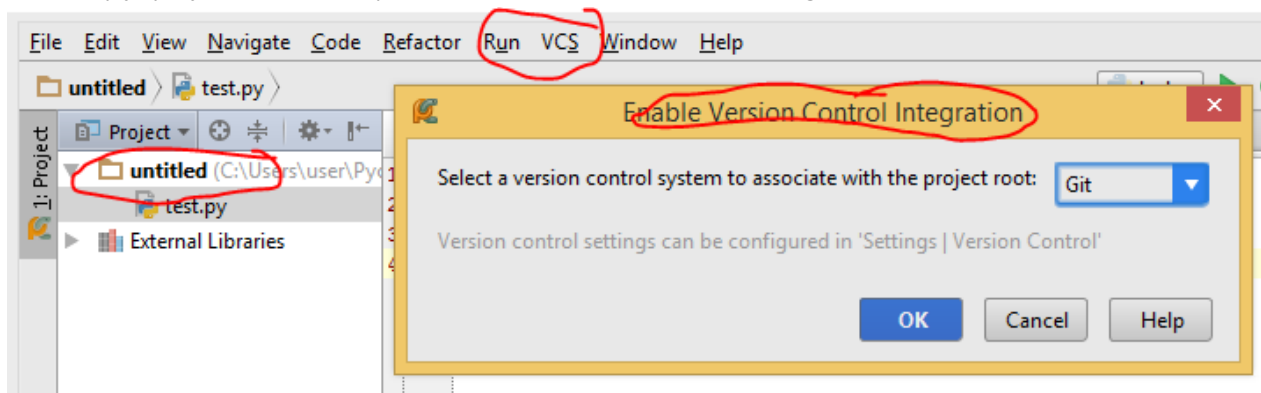
Hieronder een stappenplan om ervoor te zorgen dat je op een goede manier met elkaar kunt samenwerken aan een bestand met broncode.

6.1.3 Opdracht: Werken met Git

1. Ga naar <http://git-scm.com/download/win> en installeer GIT. Accepteer hierbij alle default instellingen.
2. Zorg er nu voor dat je GIT vanuit PyCharm kunt gebruiken: File → Settings → Version Control (klik op de driehoek) → Git.
3. Geef het goede pad naar git.exe op en klik op "Test" en daarna op "OK".



Klik nu op je project en daarna op VCS → "Enable Version Control Integration". Kies "Git" en daarna "OK".



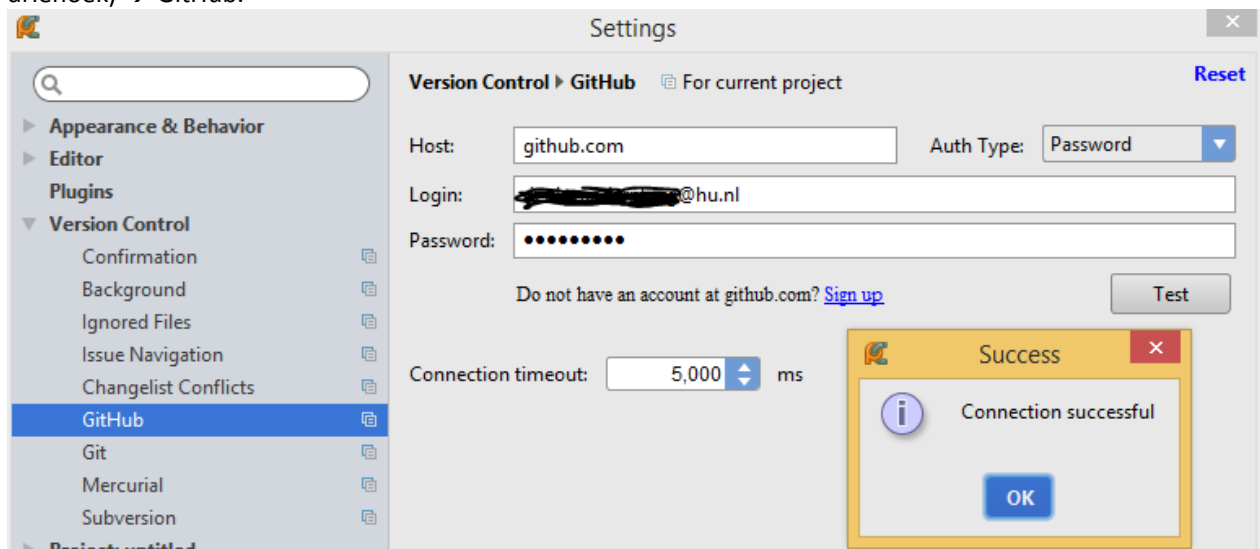
4. Zorg ervoor dat je source code terecht komt in GIT. Heel globaal is de volgorde:
 - a) Maak een nieuwe file aan in je project. Dit is niet nodig als je al een bestaand script hebt.
 - b) Voeg de file vervolgens toe aan Git. Klik hiervoor met de rechtermuisknop op de file → Git → + Add

- c) De file is nu onder “source control”. Ga door met het maken van je script. Ben je klaar?
- d) Klik hiervoor met de rechtermuisknop op de file → Git → Commit file... Hierdoor wordt je bestand in de lokale Git repository opgeslagen. Zorg bij iedere commit voor een logische “commit message” als commentaarregel. Bij de eerste commit wordt je gevraagd om een Git User Name en Mail adres. Kies een logische waarde.
- e) Je source is nu opgenomen in de lokale “database”.
- f) Pas wat aan in je script (regel erbij / regel eraf / regel aanpassen). Vergelijk je bestand nu met de versie dit is opgeslagen in Git (Git→Compare with latest...)
- g) Als je tevreden bent met de code kan je deze voorzien van een Tag (Git→Tag...) Dat is een label en dit wordt vaak gebruikt om een versie van de software aan te geven bv “1.0”.
- h) Zie je hoe je terug kan naar een eerder opgeslagen versie?

6.1.4 Opdracht: Werken met GitHub

De lokale Git database is een goede oplossing, maar soms wil je met meerdere mensen samen werken of gewoon een backup hebben op een ander systeem. Dit kan ook met Git, maar we kiezen nu voor opslag in de cloud door middel van GitHub.

- 5. Maak een account aan op Github: <https://github.com/join>
- 6. Zorg er nu voor dat je GitHub vanuit PyCharm kunt gebruiken: File → Settings → Version Control (klik op de driehoek) → GitHub.



- 7. Nu alleen nog je project publiceren op GitHub: Vanuit PyCharm: VCS → Import into Version Control → Share Project on GitHub. Geef de repository een logische naam.
- 8. Kijk op <https://github.com> Kan je daar je code terugvinden?
- 9. Je kan nu op elk moment je wijzigingen “committen” (lokaal opslaan) of “committen en pushen” (lokaal en ook op GitHub bewaren).
- 10. Ten slotte: Wil je de code van iemand anders binnenhalen (“clonen”). Doe dat via VCS → Checkout from Version Control → GitHub...

6.2 College 1

6.2.1 Opdracht: Namespaces

Corrigeer de onderstaande code. Er wordt verkeerd gebruik gemaakt van namespaces...

```
def verdubbel(b):  
    return b+b  
print(verdubbel(7))  
print(b)
```

```
import * from math  
print(math.pi)
```

```
print(f(3))  
def f(x):  
    return 2*x+1
```

```
print (time.strftime("%H:%M:%S"))
```

```
from time import *  
print (time.strftime("%H:%M:%S"))
```

```
a=3  
def f():  
    global a  
    a=9  
print("waarom is a niet 9?", a)
```

6.2.2 Opdracht: Object Oriented Programming

Maak een klasse "Train"

1. Bedenk Attributes voor deze klasse:
Fields (=velden)
Methods (=functies)
2. Maak een Constructor die zorgt voor initialisatie van de velden.
3. Maak een functie die de juiste "snelheid" regelt (versnellen/vertragen).
4. Maak een functie die het object kan printen.

6.3 Weekopdracht

6.3.1 Opdracht: Object-georiënteerd programmeren (PythonIntroduction)

Maak vanuit het PythonIntroduction project: "lesson8" (Classes and objects)

6.3.2 Opdracht: Object-georiënteerd programmeren (OOP)

Je hebt gehoord dat Object-georiënteerd programmeren (OOP) veel voordelen biedt en dat OOP door Python ondersteund wordt. We gaan dit nu toepassen bij het gebruik van object georiënteerd programmeren. Het doel is om de OV-chipkaart als een object op te nemen in je programma. Het eindresultaat moet je met een tweetal Python bestanden laten zien.

De OV-chipkaart die je nodig hebt om met de trein te kunnen reizen wordt geleverd door Trans Link Systems. Om een kaart aan te vragen moet je onderstaande informatie opvoeren.



FIGUUR 10: BRON: [HTTPS://WWW.OV-CHIPKAART.NL/AANVRAGEN/UITLEG-PERSONLIJKE-OVCHIPKAART/PERSONLIJKEOVCHIPKAART/KAARTAANVRAGEN/](https://www.ov-chipkaart.nl/aanvragen/uitleg-persoonlijke-ovchipkaart/persoonlijkeovchipkaart/kaartaanvragen/)

1. Maak in Python een minimale klasse "Smartcard" met een docstring waarin je de klasse globaal omschrijft. Sla deze klasse op in een apart bestand met de naam smartcard.py
1. Geef de klasse een constructor (`__init__` functie) waarmee de gegevens uit figuur 1 worden vastgelegd in het object.
2. Zorg voor een uitbreiding van de constructor zodat de "creation-date" wordt toegevoegd aan het object. Gebruik hierbij bijvoorbeeld de module "datetime".
3. Maak vanuit een tweede python bestand "my_script.py" twee nieuwe persoonlijke smartcards aan. Importeer hiervoor in de eerste regel van je script de klasse "smartcard". Gebruik hierbij de constructor uit de vorige opdracht.
4. Maak binnen de klasse een methode "check_name" waarmee je controleert of het totale aantal karakters van voorletters+tussenvoegsel(s)+achternaam niet groter is dan 26. Als de naam geldig is geeft de methode "True" terug en anders "False". Roep de functie aan voor iedere user vanuit my_script.py en print het resultaat op het scherm.
5. Maak binnen de klasse een methode "reload" waarmee je geld kan toevoegen aan de kaart.

6. Maak ook een methode “withdraw” waarmee je geld afboekt van de kaart. De methode geeft “None” terug als er een negatief saldo zou kunnen ontstaan.
7. Geef elke methode een logische docstring waarin je de functie beschrijft en aangeeft welke parameters als input worden gegeven en wat de functie teruggeeft.
8. Zorg ervoor dat elke smartcard een uniek nummer krijgt door hiervoor een klasse variabele te gebruiken.
9. Zorg ervoor dat het printen van een klant object de naam, het adres en de woonplaats toont van deze klant. Gebruik hiervoor de speciale functie `__str__`.
10. Maak nu het hoofdsript. Dit script kent de volgende flow:
 - a. Importeer je klasse “smardcard”.
 - b. Vraag de gebruiker om input. De input gebruik je om een nieuwe smartcard te initialiseren. Het object dat je hiervoor gebruikt is `personal_ov_1`.
 - c. Vraag de gebruiker om input. De input gebruik je om een tweede smartcard te initialiseren. Het object dat je hiervoor gebruikt is `personal_ov_2`.
 - d. Zorg ervoor dat op beide kaarten 10 euro wordt gestort.
 - e. Haal van de eerste card nu 5 euro af.
 - f. Haal van de tweede card nu 15 euro af (en toon een logische foutmelding).
 - g. Print ten slotte beide objecten: `print(personal_ov_1)` en zorg ervoor dat alle velden van het object worden afgebeeld op het scherm.

7 Resources

- Introduction to Computing Using Python, L. Perkovic
- Byte of Python, Swaroop C.H.