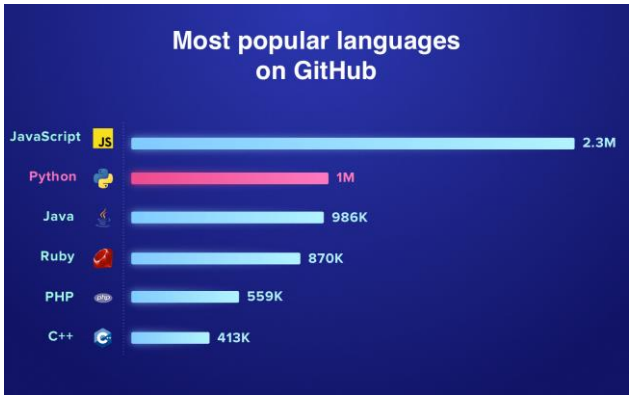
# Python 2

INLEIDING




Python is één van de meest bekende talen op Github. Alleen javaScript is nog populairder. Dat zegt iets over de veelzijdigheid van Python en het is daarmee een goede taal om te leren. Als een taal zó populair is, is er ook veel vraag naar software developers die er iets van weten.
Je zult merken dat Python overeenkomsten heeft met javaScript en PHP. Maar meestal is de syntax nèt even anders.

**NB**: Python is een levende taal, wat betekent dat er nieuwe versies komen met nieuwe mogelijkheden en soms zelf ook bugfixes in de interpreter zelf. Weet dat er op het internet nog veel Python versie 2 code staat. Python 2 ontwikkeling is gestopt met versie 2.7 in 2020. Versie 2 code werkt niet zondermeer in de versie 3 waar jullie mee werken. (ik heb dat wel eens gezien bij W3 schools, bijvoorbeeld). Op dit moment worden alleen versies vanaf 3.6 ondersteund. Houd daar rekening mee als je op zoek gaat naar voorbeelden op het internet. Zoek naar recente voorbeelden en probeer vast te stellen of het Python versie 3.6 of nieuwer is.

Deze modules blijven relatief aan de oppervlakte van Python. Dat kan ook niet anders. Bedenk dat het jaren van leren en toepassen kost om een programmeertaal ècht te beheersen.
Deze tweede Python module gaat dieper in op eerdere stof en voegt nieuwe elementen toe.
Het is een module met relatief veel tekst.
Neem de tijd de tekst goed te lezen en tot je door te laten dringen. Er is nu eenmaal veel kennis over te dragen bij het leren van een programmeertaal. Dat verandert niet, hoe die lessen ook worden aangeboden. Dingen leren kost nu eenmaal inspanning en dat maakt het tegelijkertijd de moeite waard. Je leert dan iets dat niet iedereen kan!
Normaal gesproken zou Python in de klas worden besproken, waar ook de mogelijkheid om vragen te stellen en om hulp te vragen makkelijker is dan nu.

Je weet: programmeren leer je door het te doen! Ik wil jullie dan ook uitnodigen om zelf te experimenteren met Python. Waar een voorbeeld wordt gegeven, kun je een variatie bedenken en die programmeren. Je kunt daarbij opzettelijk dingen anders doen. Soms gaat het niet goed, dan krijg je een exception (foutmelding) en soms leidt dat tot onverwachte resultaten. Dat is prima!

Wanneer het programma iets anders doet dan je verwacht, probeer dan uit te vinden wat er precies gebeurt. De Thonny IDE kan je daarbij helpen omdat die precies laat zien hoe jouw code wordt geïnterpreteerd door de Python shell. In deel 2 komt Thonny aan de orde.

Alle vragen en opdrachten in de stof hebben de bedoeling je te laten nadenken. Neem daar dan ook de tijd voor. Als iets niet meteen lukt, probeer het dan nog een keer op een andere manier. Vraag het aan klasgenoten, zoek informatie op het internet en stel een hulpvraag in Eagledev. En, last but not least, laat het probleem even met rust. Ga iets anders doen en kom er later met een frisse blik op terug.

## Inleveren van opdrachten en uitwerkingen.

- Je maakt een repo aan met dezelfde indeling als deze module.
- De opdrachten zijn heel verschillend. Soms moet je code inleveren, die kun je kwijt als een .py file in de repo en soms moet je kunnen bewijzen dat je iets hebt gedaan. Dat kan met een screenshot van (bijvoorbeeld) het Thonny window, of met een stukje tekst.
- Eventuele plaatjes (zoals screenshots) en tekst neem je op in een PDF file en die zet je ook in jouw repo op de goede plek neer.
- Naamgeving van de files is als: Python 2.2 Thonny opdracht 1.dpf

Tot slot wil ik jullie een aantal quotes voorschotelen die door èchte programmeurs zijn geschreven. Ze zijn namelijk onverkort van toepassing, ook op deze module.

1. Je moet aan de opgaves werken totdat je ze opgelost hebt. Het volstaat niet om een beetje te proberen en dan het antwoord op te zoeken. Een dergelijke aanpak is volstrekt zinloos. Je zult nooit leren programmeren als je niet nadenkt over oplossingsmethodieken, code schrijft, en code test. Als je een opgave niet kunt oplossen zelfs als je er lange tijd aan hebt gewerkt, doe je er beter aan hulp te vragen dan het antwoord op te zoeken. Het niet kunnen oplossen van een opgave betekent dat er iets in het materiaal zit dat je nog niet begrijpt, en het is belangrijk dat je ontdekt wat dat is, zodat je dat gebrek kunt verhelpen.

2. Je moet alle opgaves maken. De enige manier om te leren programmeren is te oefenen. Je zult veel code moeten schrijven om de praktijk van het programmeren te internaliseren. De paar opgaves die ik aan het einde van ieder hoofdstuk op heb genomen zijn nog niet voldoende om dat te bereiken, maar ze zijn een begin. Als je niet de moeite neemt om al die opgaves te doen, hoef je ook niet de moeite te doen om te proberen programmeren te leren.

3. Je moet de opgaves zelfstandig maken. Aan de opgaves werken in groepsverband laat één persoon leren terwijl de rest erbij zit en toekijkt. Studenten vertellen me vaak dat ze een leermethode hebben waarbij ze aan opdrachten werken in groepsverband en antwoorden bediscussiëren. Dat werkt misschien voor het analyseren van teksten en het opzetten van experimenten, maar werkt meestal niet voor coderen. Toekijken hoe iemand anders code schrijft leert je erg weinig over het schrijven van code. Je moet zelf code schrijven.

Bron: De Programmeursleerling: Leren coderen met Python 3  Pieter Spronck