Programmeeropdrachten

# Opdracht 1

print "H H I"

print "H H"

print "H H I"

print "HHHHH I"

print "H H I"

print "H H I"

print "H H I"

print 1 + 1

print "H H I"

print "H H"

print "H H I"

print "HHHHH I"

print "H H I"

print "H H I"

print "H H I"

De bovenstaande code bevat duplicated code. Herschrijf de bovenstaande code, pas een functie toe om de duplicated code verwijderen.

# Opdracht 2

Schrijf een functie genaamd printGrootste, welke drie getallen als parameters accepteert en vervolgens d.m.v. if-statements het grootste bepaalt en uitprint. En schrijf code die de functie test/aanroept. (Dit wordt ook wel aanroepende code of client-code genoemd.)

# Opdracht 3

Het nadeel van de functie (aangeroepen code) uit de vorige opdracht is dat deze direct het grootste getal uitprint, zonder dat de aanroepende code het getal in handen krijgt: de functie stuurt het grootste getal namelijk meteen naar het beeldscherm. Dit wordt ook wel een side-effect van een functie genoemd: de functie doet \*iets\* meer dan alleen het simpelweg berekenen van het grootste getal.

De aanroepende code wilt (om wat voor reden dan ook) weten wat 2 keer het grootste getal is, en vervolgens wat 3 keer het grootste getal is. Herschrijf de code dusdanig dat de functie getal teruggeeft (return), en laat de aanroepende code dit vervolgens maal twee en apart nog een maal drie doen en uitprinten.

# Opdracht 4

Gegeven de bovenstaande willekeurig verzonnen formule, maak een functie van deze formule, waarbij de faculteit in een aparte functie wordt berekend, en de macht 3 ook weer in een aparte functie wordt berekend. Beperkingen: je mag geen ingebouwde faculteit functies van Python gebruiken. Ook mag je niet de macht/power functie of operator (^) van Python gebruiken.

Gedetailleerde uitleg:

is gelijk aan x \* x \* x. Omdat dit zowel voor x als y als z gedaan moet worden, en we in deze opdracht maximaal code hergebruik willen toepassen, mag je niet zoiets doen: x\*x\*x\*y\*y\*y\*z\*z\*z, maar moet je dus voor het macht 3 gedeelte een aparte functie schrijven, waardoor het te vervangen is door zoiets: macht3(x) \* macht3(y) \* macht3(z).

x! staat voor de faculteit van x. Zie <https://nl.wikipedia.org/wiki/Faculteit_%28wiskunde%29> voor meer info over de faculteit van een getal. In het kort.  
0! = 1

1! = 1

2! = 1 \* 2  
3! = 1 \* 2 \* 3

4! = 1 \* 2 \* 3 \* 4

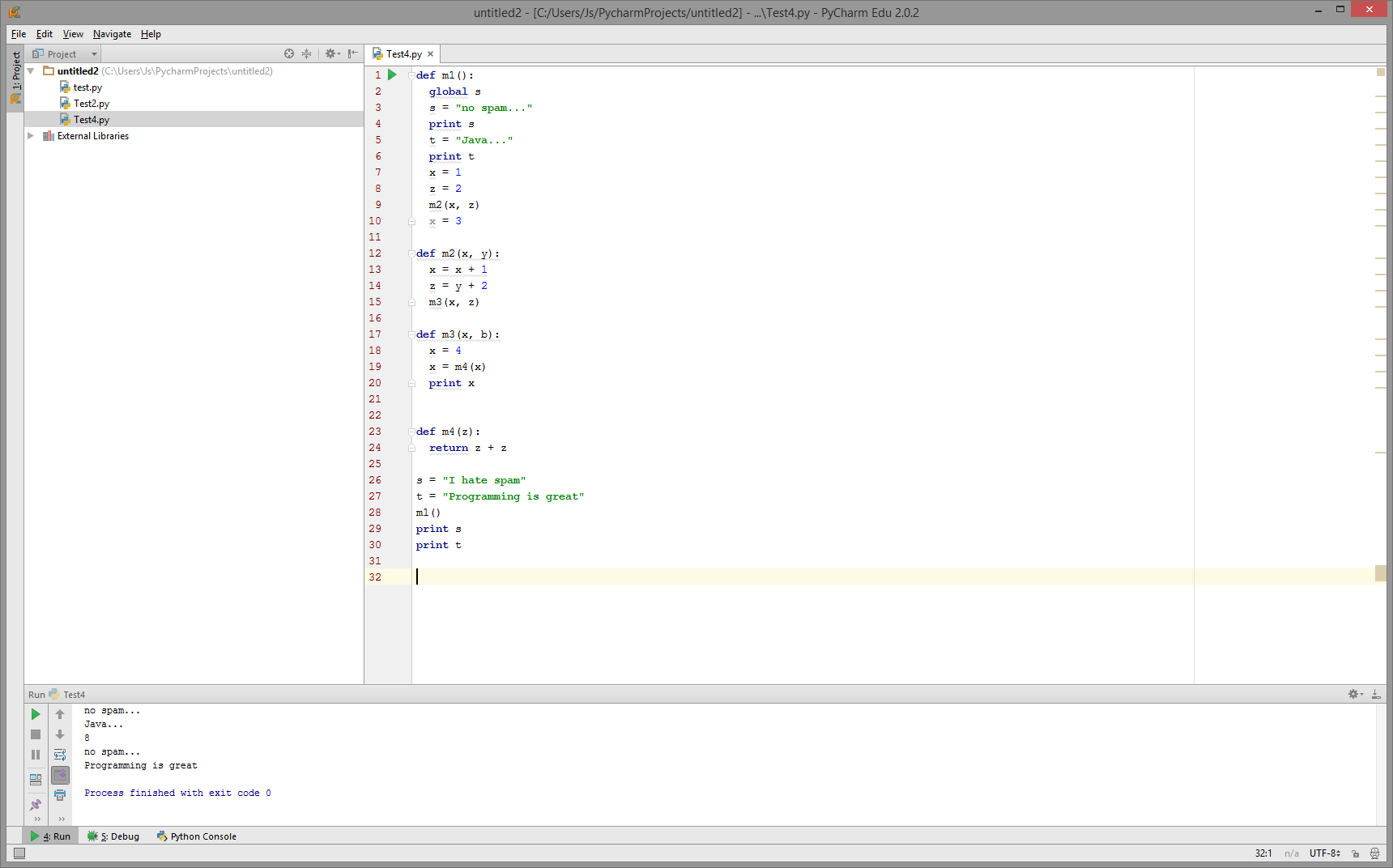
Etc.

Omdat faculteit meerdere keren in de gegeven formule voorkomt, moet je ook voor de faculteit een aparte functie schrijven. Je zal loops moeten gebruiken om dit voor elkaar te krijgen.

Uiteindelijk krijg je dus 3 functies:

1. één voor de gegeven formule, die x, y en z als parameters accepteert en als resultaat (return) de oplossing van de formule terug geeft
2. een formule voor de macht3, die dus één parameter heeft, en een getal terug geeft.
3. Een formule voor de faculteit, die dus ook één parameter heeft, en een getal terug geeft.

# Opdracht 5



D.m.v. een call stack kan Python intern (op de achtergrond) tijdens het uitvoeren van code bijhouden wat:

* Wat de functie is die op dat moment wordt uitgevoerd, en (indien aanwezig) vanuit welke vorige functie de huidige functie is aangeroepen, en hetzelfde geldtf voor de functie daarvoor en daarvoor etc.
* Welke variabelen (zowel globaal als lokaal als parameters) er dan beschikbaar zijn, inclusief de bijbehorende waarde.

De call stack kan continue veranderen bij elke volgende instructie die Python uitvoert. De call stack op een bepaald moment is dus een moment opname van wat Python op dat moment ziet.

De bovenstaande code wordt uitgevoerd, en Python belandt op een gegeven moment op de volgende regelnummers. Schrijf per regelnummer op wat de bijbehorende callstack is: 8, 10, 15, 24, 30 (De call stack moet gemaakt worden VOORDAT de regel wordt uitgevoerd.)

# Opdracht 6

Schrijf een functie genaamd print(x,y) die de getallen van x t/m y uitprint. De functie moet geschreven worden d.m.v. een recursie. Je mag geen loops gebruiken, en ook geen ingebouwde python functies anders dan de print functie.

# LIST OPDRACHT REGELS

Let op, bij het maken van onderstaande list opdrachten zijn alleen de volgende operaties toegestaan, tenzij anders aangegeven:

* Alles met []
  + woorden = []
  + print woorden[0]
  + woorden[0] = “nieuwe tekst”
* Elementaire operaties:
  + toevoegen: woorden.append(...)
* Niet toegestaan (tenzij aangegeven):
  + met name ingebouwde sortering

# Opdracht 7

Schrijf een programma dat aan de gebruiker vraagt om woorden in te voeren, totdat deze “stop” invoert, en vervolgens in omgekeerde afdruk volgorde weer uitprint.

# Opdracht 8

Schrijf een functie die als parameter een list van getallen accepteert, en vervolgens het volgende uitprint:

* de som (optelling) van alle getallen
* het product (vermenigvuldiging) van alle getallen
* het gemiddelde (som / aantal)
* het midden (het middelste getal. Indien geen precies midden is, dan moet het gemiddelde van de twee getallen in het midden genomen worden)

Tip: indien de lengte modulo 2 gelijk is aan 0, dan zijn er twee getallen in het midden, anders één.

Opmerking: indien er bijvoorbeeld staat: schrijf een functie die X accepteert, dan hoef je er niet voor te zorgen dat het Y niet accepteert. (Al hoewel het dat laatste meestal wel netter is.)

# Opdracht 9

Schrijf een functie die als parameter een list van getallen accepteert, en vervolgens als return value het grootste getal teruggeeft.

# Opdracht 10

Schrijf een functie die als parameters twee lists van getallen accepteert. Je mag er vanuit gaan de alleen de getallen van 0 t/m 10 voorkomen in deze lists. Getallen kunnen mogelijk wel meerdere keren voorkomen in een list. Vervolgens moet de functie een nieuwe list teruggeven, welke een lengte van 10 moet hebben, en waarbij index 0 aangeeft hoevaak 0 voorkwam in de andere lists, index 1 aangeeft hoevaak 1 voorkwam, index 2 aangeeft hoevaak 2 voorkwam, enz.

# Opdracht 11

Schrijf een functie die als parameter twee lists van getallen accepteert, met beiden eenzelfde lengte. Vervolgens moet de functie de getallen uit de ene list vermenigvuldigen met de getallen uit de andere list, en deze plaatsen in een nieuwe list:

* getal op index 0 van list1 vermendigvuldigen met getal op index 0 van list2, en plaatsen op index 0 van een nieuwe list
* getal op index 1 van list1 vermendigvuldigen met getal op index 1 van list2, en plaatsen op index 1 van een nieuwe list
* getal op index 2 van list1 vermendigvuldigen met getal op index 2 van list2, en plaatsen op index 2 van een nieuwe list
* enz

# Opdracht 12

Maak het spel tik-tak-tor na met een speelraster van 5 bij 5. De spelers worden om de beurt gevraagd om cooridinaten in te voeren waar hun zet wordt geplaatst. Controlleer bij elke plaatsing of de coordinaat wel vrij is. En print na elke zet opnieuw het speelraster uit, met de nieuwe zet daar aan toegevoegd. Je hoeft echter niet te controlleren of een speler gewonnen of verloren heeft, dan moeten de spelers zelf doen.

De stappen:

1. Het spel begint met een leeg raster, print deze uit.
2. Vraag aan speler 1 de x coordinaat
3. Vraag aan speler 1 de y coordinaat
4. Plaats de zet in het raster en print deze uit
5. Ga terug naar stap 2, alleen dan voor de andere speler.

# Opdracht 13

Maak een applicatie waar de gebruiker cijfer gegevens van een student kan invoeren. Deze gegevens bestaan uit: naam, vakcode, cijfer.

De cijfers moeten achter elkaar in een loop worden ingevoerd, totdat de gebruiker “stop” invoert. Daarna moet de gebruiker een vakcode invoeren, waarna het programma het gemiddelde cijfer van dat vak moet uitprinten.

Gebruik een 2-dimensionale list om de gegevens te verzamelen.

# Opdracht 14

Hetzelfde als opdracht 13, alleen dient i.p.v. een 2-dimensionale list, class/objecten te worden gebruikt. Bepaal zelf de naam van de class. Zorg ervoor dat de gegevens in het object door middel van een constructor worden geforceerd. In opdracht 13 mogen zowel float als integers worden gebruikt als cijfer, hetzelfde geldt voor de constructor van opdracht 14. Echter mogen de eindcijfers alleen afgeronde cijfers zijn. (Afronden kan met de round() functie.)

Plaats daarom ook een eindcijfer() method in de class, zodat bij het stoppen dmv “stop” invoeren, het gemiddelde eindcijfer wordt genomen (berekent door de eindcijfer() method), i.p.v. de niet-afgeronde cijfers.

# Eigen LIST (Node/Empty) OPDRACHT REGELS

In de onderstaande opdrachten mag geen gebruik worden gemaakt van de ingebouwde List van Python. Je mag alleen je eigen List implementatie gebruiken.

# Opdracht 15

Schrijf een method genaamd printAll in de Node class class die de volledige node (inclusief de tail) uitprint, van voor naar achteren.

De test code zou er zo uit kunnen zien:

mijnList = Node(“a”, Node(“b”, Node(“c”, Empty)))

mijnList.printAll() # screenOutput: a,b,c,

# Opdracht 16

Schrijf een method genaamd size() in de Node class. De method moet het aantal values teruggeven.

De test code zou er zo uit kunnen zien:

mijnList = Node(“a”, Node(“b”, Node(“c”, Empty)))

print mijnList.size == 3 # screenOutput: True

# Opdracht 17

Schrijf een method genaamd get(index) in de Node class. De method moet m.b.v. de index parameter de value van de node op die index teruggeven als return.

De test code zou er zo uit kunnen zien:

mijnList = Node(“a”, Node(“b”, Node(“c”, Empty)))

print mijnList.get(2) == “c” # screenOutput: True

# Opdracht 18

Schrijf een method genaamd printAllReverse in de Node class. De method is vergelijkbaar met de printAll method, alleen moeten de elementen in omgekeerde volgorde worden uitgeprint.

De test code zou er zo uit kunnen zien:

mijnList = Node(“a”, Node(“b”, Node(“c”, Empty)))

mijnList.printAllReverse() #screenOutput: c,b,a

# Opdracht 19

Schrijf een method genaamd delete(index) in de Node class. De method moet m.b.v. de index parameter de value van de node op die index verwijderen en teruggeven als return. Door het verwijderen ontstaat er een gat, maar dit gat moet worden gedicht, zodat de elementen weer aaneengesloten zijn, m.a.w. er ontstaat een verschuiving om het gat te dichten.

De test code zou er zo uit kunnen zien:

mijnList = Node(“a”, Node(“b”, Node(“c”, Node(“d”, Empty))))

mijnList.printAll() # screenOutput: a,b,c,d,

opgeruimd = mijnList.delete(2)

print opgeruimd == “c” # screenOutput: True

mijnList.printAll # screenOutput: a,b,d,

# Opdracht 20

Als je met collecties/lists werkt, komt het regelmatig voor dat je een transformatie moet maken van de list. Bijvoorbeeld. Stel je hebt een list van getallen, en je moet elk getal in de list verdubbelen (vermenigvuldigen met twee), dan heb je dus voor elk getal in de list een transformatie van x naar x\*2. Op dezelfde wijze zou je met een list van student objecten (waarbij elke student bijv. een naam, leeftijd en woonplaats heeft), de list van studenten kunnen transformeren naar een list met daarin alleen de woonplaats (String) van elke student. Bij een transformatie blijft de lengte van de list hetzelfde, alleen de waarde verandert (meestal), en het type (int, float, string, object etc) kan ook veranderen.

Schrijf een functie genaamd “transform” die als eerste argument/parameter een list accepteert, en als tweede parameter een lambda expressie. De lambda expressie stelt een transformatie functie voor, die een object/value (bijv. een getal) omzet naar een andere value (bijv. getal \* 2). Als return moet de “transform” functie een nieuwe list teruggeven, waarbij elke element van de list parameter is omgezet naar een andere value d.m.v. de transformatie functie.

Probeer de “transform” functie uit door een list van student objecten (elk student heeft een naam, woonplaats, en leeftijd) mee te geven, waarbij de transformatie een list teruggeeft met daarin alleen de woonplaatsen. (De woonplaatsen mogen meerdere keren voorkomen in het resultaat.)

# Opdracht 21

Deze opdracht hoeft niet ingeleverd te worden.

Voer de Codecademy opdrachten 12 t/m 18 uit: <https://www.codecademy.com/en/courses/python-beginner-en-KAgt5/3/1>

# Opdracht 22

Deze opdracht hoeft niet ingeleverd te worden.

Voer de Codecademy opdrachten 1 t/m 3 uit:   
<https://www.codecademy.com/en/courses/python-beginner-en-KAgt5/0/1>

Voer de Codecademy opdrachten 10 t/m 14 uit:   
<https://www.codecademy.com/en/courses/python-beginner-en-pwmb1/2/1>

# Opdracht 23

In deze opdracht mag niet met aparte klassen worden gewerkt. Werk dus met basis data typen, zoals Map, List, Set, String, Integer, Float, boolean etc.

Maak een programma waarmee administratie medewerkers gegeven een studentnummer, de bijbehorende gegevens van de student kunnen invoeren. Deze bijbehorende gegevens bestaan uit: voornaam, achternaam, klas.

Na het invoeren van meerdere studenten, moet de medewerker gegevens van studenten kunnen opvragen, gegeven een studentnummer.

Bepaal zelf de meest geschikte collecties.