```
########################
# Opdracht 1 Variabelen #
########################

# Het print commando zorgt ervoor dat alles tussen de haakjes
# op het scherm wordt geprint. Dat kan gewoon een stukje tekst
# zijn tussen aanhalingstekens

print ("Hallo, Coderdojo!")

# Alles achter het #-teken wordt niet uitgevoerd.
# Dit heet 'Documentatie' of 'Commentaar'
# en mag ook gewoon ergens midden in een regel staan

print ("15 april 2017: Workshop Python") # kijk maar, werkt prima

# Opdracht 1a
# Druk nu maar eens op de Run knop hierboven. Had je dit verwacht?

# Opdracht 1b
# Pas de tekst eens bovenaan aan zodat je eigen naam op het scherm komt

print ("Hallo Arjen!")

# Opdracht 1c
# Met variabelen kun je iets even onthouden, om het later weer te gebruiken.
# Kun je nu hieronder met print(naam) jouw eigen naam op het scherm zetten?

naam = "Super Ninja"

print(naam)
```

```
##############################
# Opdracht 2 Types           #
##############################

naam     = "Arjen"        # een string (maak hier je naam van)
leeftijd = 40             # een geheel getal (maak hier je leeftijd van)
zakgeld  = 1.5            # een gebroken getal

# Opdracht 2a
# Strings kun je ook samenvoegen met een , of met een +
# bijvoorbeeld: "Hallo " + naam + ", welkom bij Python"
# Zorg nu eens dat de zin hierboven wordt geprint, maar dan met je eigen naam?

print ("Hallo " + naam + ", welkom bij Python")

# Opdracht 2b
# Je kunt een string ook twee (of meer) keer achter elkaar printen door * te gebruiken
# bijvoorbeeld: naam * 2
# Zorg dat je de naam-variabele gebruikt, en print op het scherm (je mag ook je eigen naam ge…
# Hallo, ik ben NinjaNinjaNinjaNinja

print ("Hallo, ik ben", naam * 4)

# Opdracht 2c
# Je kunt ook makkelijk rekenen met je leeftijd, door + - * en / te gebruiken.
# Zo'n teken heet een operator: een mooi woordt om handelingen ergens op uit te voeren.
# bijvoorbeeld: leeftijd + 10
# Hoe zou je op je scherm kunnen laten zien:
# - hoe oud je over 5 jaar bent, of hoe oud je 5 jaar geleden was?
# - hoe oud je bent als je 3x zo oud bent als nu?

print (leeftijd + 5)
print (leeftijd - 5)
print (leeftijd * 3)

# Opdracht 2d
# Tafels leren hebben we allemaal gedaan, en kunnen we nu ook heel makkelijk doen met variabe…
# We gaan nu de tafel van je leeftijd printen door 10 regels code onder elkaar te zetten.
# Kun jij de lijst afmaken zodat de hele tafel wordt geprint?
# Hoe kun je nu heel makkelijk een andere tafel printen?

getal = leeftijd
print ('De tafel van', getal)
print ('1 x', getal, '=', 1 * getal)
print ('2 x', getal, '=', 2 * getal)
print ('3 x', getal, '=', 3 * getal)
print ('4 x', getal, '=', 4 * getal)
print ('5 x', getal, '=', 5 * getal)
print ('6 x', getal, '=', 6 * getal)
print ('7 x', getal, '=', 7 * getal)
print ('8 x', getal, '=', 8 * getal)
print ('9 x', getal, '=', 9 * getal)
print ('10 x', getal, '=', 10 * getal)

# Opdracht 2e
# Gebroken getallen zijn bijvoorbeeld halven: 1,50 eur zakgeld per week.
# Ook daarmee kun je net zo makkelijk rekenen als met hele getallen.
# Kun je op je scherm laten zien hoeveel geld je na 13 weken hebt gespaard?
# Kun je op je scherm laten zien hoeveel weken je moet sparen voordat je 54 euro hebt gespaar…
# Het beginnetje is er, jij moet het nog even afmaken! (Haal je ook even het # weg hieronder?)

print ("Na 13 weken heb ik", zakgeld * 13, "euro zakgeld bij elkaar gespaard")
print ("Ik moet", 54 / zakgeld, "weken sparen voordat ik 54 euro heb gespaard")

# Opdracht 2f
# Lijstjes kun je makkelijk gebruiken om meer van hetzelfde bij elkaar te houden.
# Dit geven we aan met rechte haken, en komma's ertussen.
# bijvoorbeeld:

mentoren = ["Arjen", "Dick", "Dion", "Nedim", "Patricia"]

# De eerste uit de lijst is mentoren[0], de tweede mentoren[1], enzovoort.
# Even wennen: je ziet dat een computer bij het tellen begint bij plaats 0!
# Kun je onderstaande opdrachten uitvoeren? Snap je wat er gebeurt?

print ("Alle mentoren:", mentoren)
print ("De eerste mentor is", mentoren[0])
print ("De andere mentoren:", mentoren[1:])
```

```
# Opdracht 2g
# Kun je, door goed te tellen, Nedim en Patricia op het scherm printen? Het beginnetje staat …

print (mentoren[3])
print (mentoren[4])

####################
# Opdracht 2 extra #
####################

# Je kunt ook een deel van de string opvragen met behulp van rechte haken:
# naam[0] is de eerste letter uit de string, naam[1] is de tweede letter, enzovoorts.
# Ook een deel van de naam kun je zo gebruiken met een dubbele punt:
# naam[:3] zijn letters op plaatsen 0, 1 en 2 uit je naam (de eerste 3 dus)
# naam[2:] zijn de tweede tot en met de laatste letter

# Opdracht 2h
# Voer de volgende opdrachten eens uit. Wat gebeurt er denk je?
# Kun je de zinnen kloppend maken door met rechte haken te werken?

print ("Mijn naam begint met de letter", naam[0])
print ("De andere letters uit mijn naam zijn", naam[1:])

# Opdracht 2i
# Een lastige: Kun je nu ook de tweede, derde en vierde letter uit je naam printen?

print(naam[1:4])
```

```
############################
# Opdracht 3 Invoer        #
############################

# Opdracht 3a
# Een functie is een taak die de computer uit moet voeren, en kun je herkennen aan de haakjes…
# print() is ook zo'n taak: laat iets op het scherm zien.
# Een andere taak die de computer kan uitvoeren is om de gebruiker iets te vragen.
# Dit gebeurt ook met een functie: input("Dit stukje tekst komt op het scherm als vraag ").
# Kun jij eens aan de gebruiker vragen hoe oud hij is?
# Kun je dit ook printen op het scherm (zie bijvoorbeeld opdracht 2d)?

vraag = input ("Hoe oud ben je? ")
print ("Aha, je bent dus ", vraag)

# Opdracht 3b
# Alles wat de gebruiker invoert is voor de computer een string. Dat wordt dus moeilijk reken…
# Een computer kan wel veel, maar niet uit zichzelf getallen omzetten naar tekst en omgekeerd.
# Willen we van de leeftijd weer een getal maken, dan hebben we weer een functie nodig: int()
# De functie int() maakt van hetgene tussen de haakjes weer een getal.
# Nu de opdracht net als 2c:
# Kun je nu de leeftijd vragen aan de gebruiker, en daarna printen hoe oud hij over 10 jaar i…
# Kun je ook printen hoe oud hij is als hij 2x zo oud is als nu?

leeftijd = int(vraag)
print (leeftijd + 10)  # Over tien jaar ben ik?
print (leeftijd * 2)  # hoe oud ben als ik twee keer zou oud ben als nu?

# Opdracht 3c
# Hierboven zouden we eigenlijk willen printen:
# print ("Over tien jaar ben je " + leeftijd + 10)
# Maar hier moeten we het getal weer omzetten naar tekst, dat kan de computer niet voor ons.
# Ook daar is gelukkig weer een functie voor: str()
# Deze taak maakt van hetgene tussen de haakjes een string.
# Kun je nu ook mooie zinnen maken met de leeftijden in opracht 3b?

print ("Over tien jaar ben je " + str(leeftijd + 10))
print ("Als je 2x zo oud ben als nu, ben je " + str(leeftijd * 2))

# Opdracht 3d
# We weten nu wat een string is (touwtje tekst), een int (een heel getal), en een float (gebr…
# En we hebben gezien hoe je operators kunt gebruiken (dus + - * /) om handelingen
# uit te voeren op types, maar dat dat alleen maar kan als alles van hetzelfde type is.
# Als laatste hebben we geleerd hoe je het ene type om kan zetten naar het andere.
# Kijk eens naar onderstaande regels. Kun je voorspellen wat er gaat gebeuren?
# controleer het door het 1 voor 1 uit te voeren, en verbeteren als dat nodig is.

getal = 13                      # we beginnen met een heel getal
print (float (getal))               # Wat zou eruit komen als ik een int omzet naar float
print ("Het getal is " + str(getal))    # Gaat dit goed? Hoe kun je dat verbeteren?
helft = getal / 2
print (helft)                       # getal was een heel getal, maar is nu gebroken!
print (int (helft))                 # wat als je een gebroken getal weer heel maakt?
```

```
##############################
# Opdracht 4 Lussen          #
##############################

# Opdracht 4a
# Goed, weet je de tafel uit opdracht 2e nog?
# De tien regels code zijn veel van hetzelfde, dat kunnen we met een lus oplossen.
# Hiervoor hebben we een lijst getallen nodig van 1 tot 11 (dus 1 tot en met 10)
# Ook hiervoor is een functie: range(1, 5) is een lijstje getallen van 1 tot en met 4
# Kun je nu met behulp van een lus de tafel maken? (en de tafel van 24240089, lukt dat ook?)
# Hint: aantal, "x", getal, "=", aantal * getal

getal = 40
for aantal in range(1, 11):
    print (aantal, "x", getal, "=", aantal * getal)

# Opdracht 4b
# Nu kunnen we ook eens aan de gebruiker vragen welke tafel hij wil!
# Dat kan met bijvoorbeeld input ("Welke tafel wil je zien? ")
# Kun je met input() en met int() nu de tafel laten zien die de gebruiker opgeeft?

tafel = input ("Welke tafel wil je zien? ")
getal = int (tafel)
for aantal in range(1, 11):
    print (aantal, "x", getal, "=", aantal * getal)

# Opdracht 4c
# Stel nou eens dat een honderdtafel niet van 1 tot 10 ging, maar van 1 tot 100
# Kun je dan eens de honderdtafel van 23 programmeren?

getal = 23
for aantal in range(1, 101):
    print (aantal, "x", getal, "=", aantal * getal)

# Opdracht 4d
# We hebben de lijst van mentoren nog uit opdracht 2g
mentoren = ["Arjen", "Dick", "Dion", "Nedim", "Patricia"]
# We kunnen nu met behulp van nummers en een lus ook de namen onder elkaar printen:
# for nummer in [0, 1, 2, 3, 4]: # doe iets
# Wat moet je er nu achter zetten om iets uit de lijst van mentoren te printen?

for nummer in [0, 1, 2, 3, 4]: print (mentoren[nummer])

# Opdracht 4e
# Eigenlijk kan het nog wel ietsje korter, we hebben de nummers niet eens nodig.
# Probeer de regel hieronder eens uit te voeren. Kun je verklaren wat er gebeurt?

for mentor in mentoren: print (mentor)

# Opdracht 4f
# In opdracht 4b en 4c heb je een honderdtafel gemaakt, en de gebruiker gevraagd welke tafel …
# Zou je nu de gebruiker 2 vragen willen stellen?
# 1. Welke tafel wil je zien?
# 2. Tot hoever moet de tafel gaan?
# en dan de code maken die deze tafel dan op het scherm print?
# Hint: Tot hoever moet je in range () gebruiken, maar dat is nog niet totEnMet!

tafel = input ("Welke tafel wil je zien? ")
getal = int (tafel)
hoever = input ("Tot hoever moet de tafel gaan? ")
totEnMet = 1 + int (hoever)
for aantal in range(1, totEnMet):
    print (aantal, "x", getal, "=", aantal * getal)
```

```
############################
# Opdracht 5 Beslissen     #
############################

# Opdracht 5a
# Je kunt aan een computer vragen of twee dingen hetzelfde zijn of niet, of groter of kleiner.
# Dit kan met == (gelijk), != (niet gelijk), > (groter), < kleiner
# De computer antwoord dan in het engels met True (waar) of False (niet waar)
# Als je naar de volgende regels kijkt, kun je dan bedenken of de computer waar of niet waar …
# Oh, en ja, je leert meteen een nieuwe functie:
# je kunt van een string vragen of hij ergens mee begint, of eindigt!

getal = 2
print (getal == 2) # Waar
print (getal == 3) # Niet waar
print (getal < 3)  # Waar

string = "Hello world!"
print (string.startswith("Hello"))       # Waar
print (string.endswith("asdfasdfasdf"))  # Niet waar

# Opdracht 5b
# De code hieronder laat weten of een getal deelbaar is door twee.
# Dat gebeurt met het % teken. Dit krijg je meestal niet op school, maar voor programmeurs
# is het heel handig: het laat zien of er een rest-waarde is als je twee getallen deelt.
# dus 4 % 2 = 0, maar 5 % 2 = 1
# maar ook 9 % 3 = 0 en 11 % 3 = 2
# Kun je de code testen?
# Kun je het zo aanpassen zodat er wordt gekeken of een getal deelbaar is door 3?

getal = input ("Geef een getal: ")
rest = int (getal) % 3
if rest == 0: print (getal, "is deelbaar door 3.")
else:         print (getal, "is niet deelbaar door 3.")

# Opdracht 5c
# Kun je iets maken wat:
#   Eerst een woord aan de gebruiker vraagt
#   Als het woord met een C begint, dan print je "Het woord start met de C!"
#   En anders print je "Het woord start niet met de C..."
# Het kan op meerdere manieren, kun je twee manieren bedenken?

string = input ("Geef een woord: ")
if string[0] == "C": print ("Het woord start met de C!")
else:                print ("Het woord start niet met de C...")

string = input ("Geef een woord: ")
if string.startswith("C"): print ("Het woord start met de C!")
else:                      print ("Het woord start niet met de C...")

# Opdracht 5d
# Hieronder staat een lange zin, met heel veel letters. Ook staat er een tellertje op nul.
# Kun je daaronder iets maken met for en met if zodat het aantal keer de letter "e" geteld wo…
# Hint: doe iets met bijvoorbeeld for letter in langezin en met teller = teller + 1

langezin = """Je mag hier zelf een lange zin intypen. Bedenk maar iets.
Het mag zelfs op twee regels staan, want ik heb hier drie aanhalingstekens
gebruikt, en dan weet Python dat hij door moet lezen tot er weer ergens
drie aanhalingstekens staan"""
teller = 0
for letter in langezin:
   if letter == "e":
      teller = teller + 1
print ("Ik heb", teller, "maal de letter e gevonden in de zin:", langezin)


# Opdracht 5e
# Misschien voel je het al aankomen: Kun je nu de gebruiker een zin op laten geven,
# en daarna vragen welke letter hij wil tellen uit die zin? Print het antwoord op het scherm!

langezin = input ("Typ hier een zin zin:")
zoekletter = input ("Welke letter moet ik tellen?")
teller = 0
for letter in langezin:
   if letter == zoekletter:
      teller = teller + 1
print ("Ik heb", teller, "maal de letter", zoekletter, "gevonden")
```

```
############################
# Opdracht 6 Functies      #
############################

# Opdracht 6a
# We gaan een functie maken die de titel van de opdracht op het scherm zet.
# De argumenten zijn het opdrachtnummer (6), en het onderwerp van de opdracht (Fucnties)
# Op het scherm moet dit dan als volgt worden geprint:
# ===== Opdracht 6: Functies =====
# Kun jij de functie afmaken, zodat het goed op het scherm komt?

def print_opdracht_titel (opdrachtnummer, onderwerp):
    "Deze functie print de titel van een opdracht, en een lege regel ervoor"
    titel = "=" * 5 + " Opdracht " + str(opdrachtnummer) + ": " + onderwerp + " " + "=" * 5
    print ("")
    print (titel)

print_opdracht_titel (1, "Variabelen")
print_opdracht_titel (2, "Types")
print_opdracht_titel (3, "Invoer")
print_opdracht_titel (4, "Lussen")
print_opdracht_titel (5, "Beslissen")
print_opdracht_titel (6, "Functies")

# Opdracht 6b
# De functie hierboven gaan we nu zo aanpassen dat
# - er ook een regel met allemaal =-tekentjes voor en na de titel wordt geprint
# - de titel in allemaal grote letters wordt gezet
# - alle drie de regels precies evenveel tekentjes hebben
# Hint: de functie len("CoderDojo!") geeft het aantal tekens terug: 10
# Hint: de functie upper() maakt alle letters groot: onderwerp.upper()

def print_opdracht_titel_groot (opdrachtnummer, onderwerp):
    "Deze functie print de titel van een opdracht in hoofdletters, en randje eromheen"
    titel = "=" * 5 + " Opdracht " + str(opdrachtnummer) + ": " + onderwerp.upper() + " " + "…
    lengte = len(titel)
    print ("")
    print ("=" * lengte)
    print (titel)
    print ("=" * lengte)
    print ("")

print_opdracht_titel_groot (1, "Variabelen")
print_opdracht_titel_groot (2, "Types")
print_opdracht_titel_groot (3, "Invoer")
print_opdracht_titel_groot (4, "Lussen")
print_opdracht_titel_groot (5, "Beslissen")
print_opdracht_titel_groot (6, "Functies")
```

```
############################
# Opdracht 7 Functies      #
############################

# Opdracht 7a
# Hieronder staat de functie die twee getallen bij elkaar op kan tellen.
# Hoeveel argumenten heeft deze functie?
# Bedenk zelf een paar sommen om te testen of de functie het goed doet.
# Kun je de functie veranderen zodat het voor +, -, * of / werkt?

def eraf (eerste, tweede):    # Twee argumenten
    "Deze functie trekt twee getallen van elkaar af"
    return eerste - tweede

uitkomst = eraf (1234, 12345)
print (uitkomst)

# Opdracht 7b
# We gaan nu een functie maken die veel veel getallen bij elkaar op kan tellen.
# We hebben nu 1 argument: een lijst met getallen.
# Kun jij de functie maken die alle getallen uit een lijst optelt?
# Hint: we zullen een lus moeten maken die door het lijstje heen gaat!
# Hint: als we nog helemaal niets hebben opgeteld, dan is de som 0

getallenlijst = [1, 7, 3, 91, 6, 2, 34]

def tel_veel_op (lijst):
    "Deze functie telt alle getallen in een lijst op"
    som = 0
    for getal in lijst:
        som = som + getal
    return som

print (tel_veel_op (getallenlijst))

# Opdracht 7c
# Als laatste zoeken we uit een lijst van getallen het grootste getal.
# Je kunt de lijst hierboven gebruiken als startpunt, en de functie kan ook van pas komen.

def zoek_de_grootste (lijst):
    "Deze functie zoekt het grootste getal in een lijst"
    grootste = 0
    for getal in lijst:
        if getal > grootste:
            grootste = getal
    return grootste

print (zoek_de_grootste (getallenlijst))
```