1. Objectgeoriënteerd

Objectgeoriënteerd programmeren is een systeem gebaseerd op objecten die bestaan uit data/gegevens (attributen) en code (methodes) om deze gegevens te bewerken of verwerken.

In Python worden objecten gemodelleerd als instanties (elementen) van een klasse. Een klasse kan beschouwd worden als een blauwdruk die de definitie van een object bepaalt.

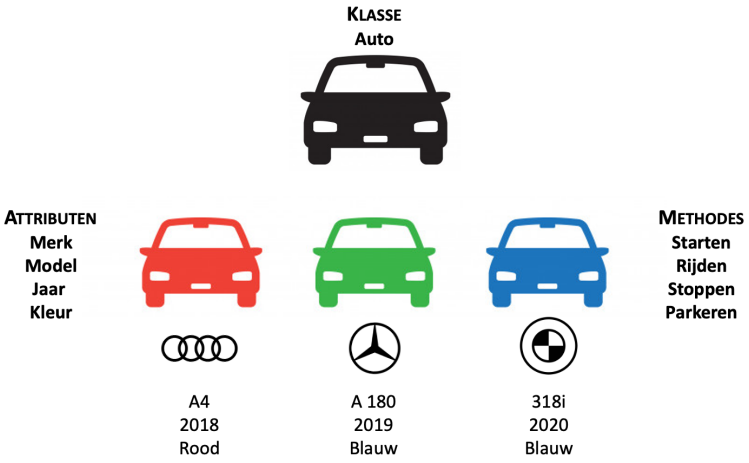
Een klasse bestaat uit:

- **Attributen** die de toestand/eigenschappen van een object vastleggen
- **Methodes** of functies die de toestand van het object aanpassen en het gedrag van een object bepalen.

Een speciale methode (de constructor) instantieert een object met zijn eigen waarden van de attributen; de toestand van een object.

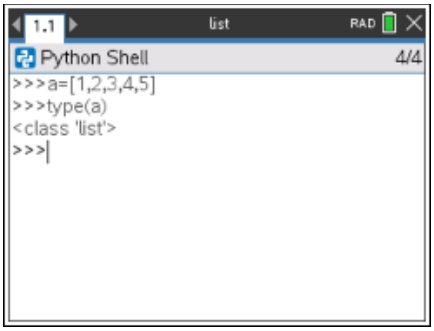
Object Oriented Programming (OOP) laat programmeurs toe hun eigen objecten en bijhorende methodes te construeren en zo hun code modulair op te bouwen.

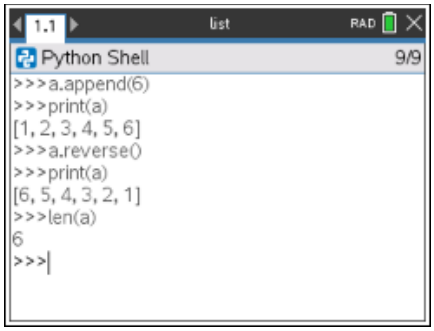
Klassen en objecten kan je vergelijken met objecten uit de ons omringende wereld zoals bv een auto.



In Python ***Everything is an object***.

In deze TI Python Bootcamp hebben we al veel gewerkt met objecten en methodes. Een voorbeeld hiervan is het data-type lijst met de methodes append() & reverse() en de speciale methodes print() & len().





We bekijken de basics van objectgeoriënteerd programmeren in Python.

2. Klassen

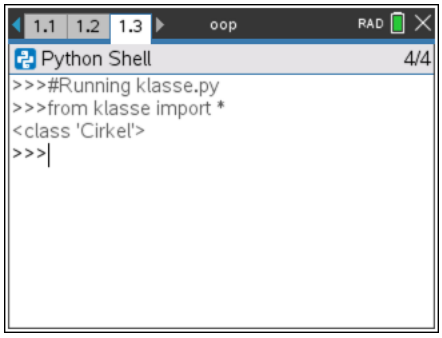
User defined objecten kunnen gecodeerd worden met het keyword class.

We illustreren de syntax van het coderen van klassen met het concept cirkel.

```
# Definitie nieuw object type
class Cirkel:
    pass

# Declaratie object van de klasse Cirkel
c=Cirkel()

print(type(x))
```



Het statement pass kan je gebruiken als een placeholder voor code. Wanneer pass wordt uitgevoerd, gebeurt er niets maar het geeft geen error. Lege code-blokken genereren namelijk een error.

Een Python-conventie geeft aan dat een klasse-naam start met een hoofdletter; net zoals we klein letters gebruiken voor variabelen.

3. Attributen

Een cirkel is analytisch volledig bepaald door zijn middelpunt en de straal. Het middelpunt en de straal gaan we coderen als de attributen(eigenschappen) van een object van de klasse Cirkel.

De syntax voor het creëren van een attribuut is b.v. self.rad = rad als onderdeel van de speciale methode __init__().

De methode __init__() initialiseert de attributen van een object en deze methode wordt steeds uitgevoerd bij het aanmaken van een object.

De algemene syntax voor het coderen van de attributen van een klasse is de volgende:

```
class Cirkel:
    def __init__(self,parameter1,parameter2):
        self.parameter1 = parameter1
        self.parameter = parameter2
```

Het keyword **self** representeerd de instantie van de klasse en wordt gebruikt om naar zichzelf te wijzen. Het is niet noodzakelijk de naam self te gebruiken maar het is een afspraak tussen Python-programmeurs.

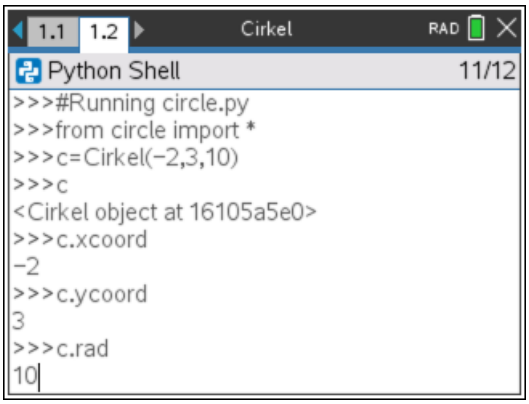
Deze syntax kan wat eigenaardig overkomen (b.v. 3x parameter1). Ook dit is een python-afspraak om telkens driemaal hetzelfde woord te gebruiken. Het is niet noodzakelijk en het zal geen error geven indien niet zo.

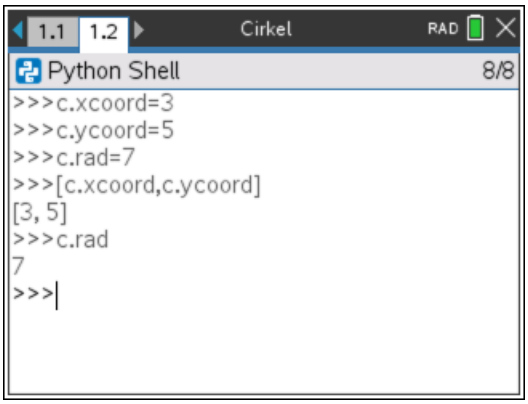
Hieronder twee dezelfde klassen voor het object Cirkel:

```
class Cirkel:
    def __init__(self,xcoord,ycoord,rad):
        self.xcoord = xcoord
        self.ycoord = ycoord
        self.rad = rad
```

```
class Cirkel:
    def __init__(self,xc,yc,straal):
        self.xcoord = xc
        self.ycoord = yc
        self.rad = straal
```

Na het declareren van een cirkel object – c = Cirkel(-3,10,2) wordt een object Cirkel aangemaakt in het geheugen. De attributen(eigenschappen) kunnen als volgt opgeroepen worden en eventueel aangepast.



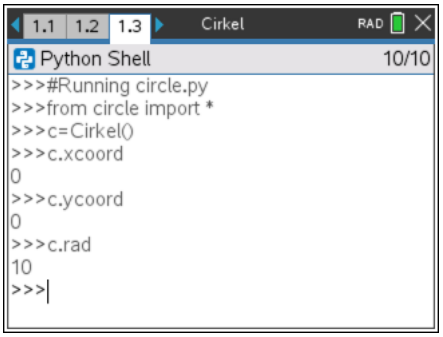


Voor de argumenten van __init__ (alsook voor andere methodes) kan als volgt aan de argumenten een standaard-waarde toegekend worden:

```
def __init__(self,xcoord=0,ycoord=0,rad=10).
```

Een Cirkel-object kan in dit geval gedeclareerd worden zonder argumenten

```
c = Cirkel().
```



Tot hiertoe hebben we de volgende terminologie van object georiënteerd programmeren geïllustreerd:

- een **klasse**, het model of de standaard van de mogelijkheden wat een object kan zijn en doen
- een **object**, de instantie van een klasse m.a.w. een concrete entiteit van een klasse
- een **instantie**, is als een object maar laten we even terug naar het concept auto:
  - een blauwdruk voor een auto-ontwerp is de klasse-beschrijving
  - alle auto's die op basis van die blauwdruk zijn vervaardigd, zijn objecten van die klasse
  - uw auto die van die blauwdruk is gemaakt, is een instantie van die klasse.

De termen instantie en object worden vaak door elkaar gebruikt en de meest voorkomende term is object.

In het volgende deel bespreken we **methodes** van een klasse.