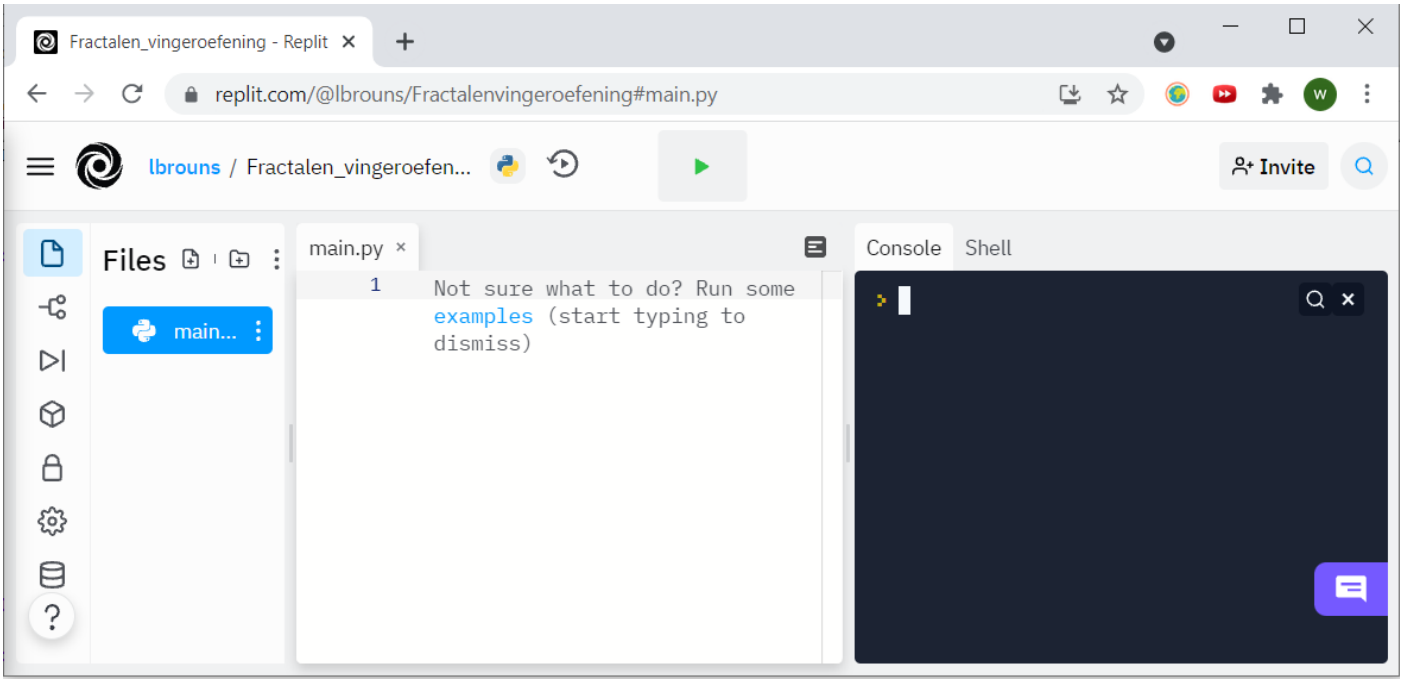
Korte python-intro

We werken met de website replit.com, daar kunnen we code schrijven en onmiddellijk laten uitvoeren. Als je ingelogd bent, krijg je het scherm te zien dat hieronder staat. In het zwarte venster rechts, de *console*, zullen we het eerste deel van de oefeningen maken. In het witte venster links, waar *main.py* boven staat, zullen we het tweede deel van de oefeningen maken.



De Python interpreter als veredelde rekenmachine

In het zwarte venster staat er een pijltje klaar (de `prompt'), waar je (python-)code kan intikken. Als je op enter drukt, zal die code onmiddellijk uitgevoerd worden. Vergelijk het met een rekenmachine: je tikt een bewerking in, en drukt op enter (of op =). In deze lesnota's wordt de prompt voorgesteld door drie pijltjes: `>>>`.

Tik in de console de bewerking `2 + 2.5` en klik op enter. Er komt:

```
>>> 2+2.5
4.5
```

Je kunt ook met variabelen werken:

```
>>> x=1.5*2
```

Dit betekent dat je een variabele `x` invoert die de waarde 3 krijgt. Let op: de variabelenaam moet in het linkerlid staan. Je mag spaties tussenvoegen, bv. `x = 1.5 * 2`. Let op kleine letters en hoofdletters; 'x' is niet hetzelfde als 'X'. Je kunt nu `x` gebruiken in berekeningen:

```
>>> x/3
1.0
```

Opm. er wordt `.0` geschreven om aan te geven dat het eigenlijk een kommagetal is (zie verder).

Vermenigvuldiging en deling hebben voorrang op optellen en aftrekken - gebruik haakjes om de volgorde te wijzigen:

```
>>> 2+3*5
17
>>> (2+3)*5
25
```

Machten met **

```
>>> 2**3
8
```

Met '_' verwijs je naar het laatste resultaat:

```
>>> _ + 1
9
```

Wiskundige constanten en functies in module `math`:

```
>>> import math
>>> math.sin(math.pi/2)
1.0
```

Om niet telkens `math.` te moeten typen (vooral interessant als de naam van de bibliotheek redelijk lang is):

```
>>> import math as m
>>> m.sin(m.pi/2)
1.0
```

Let op: het argument (hier `m.pi/2`) moet tussen haakjes staan, dus `m.sin m.pi/2` werkt niet.

Opgelet met deling: vanaf Python 3.x wordt de delingsoperator `/` gebruikt voor de reële deling. Dus de input 7/2 levert output 3.5.

```
>>> 7/2
3.5
```

In de meeste programmeertalen zou 7/2 het getal 3 opleveren: omdat 7 en 2 beide geheel zijn, wordt de gehele deling uitgevoerd (dit heeft al tot heel wat beginnersstress geleid...).

Als je in Python (vanaf versie 3.x) het afgekapte resultaat wil, moet je de operator gebruiken die in Python 3.x een gehele deling aanduidt: `//`.

```
>>> 7//2
3
```

Met `dir()` krijg je een lijst van alle constanten en functies.

Complexe getallen

Je kunt eenvoudig werken met complexe getallen. Een imaginair getal eindigt op 'j'. Opgelet: 'j' moet voorafgegaan worden door een getal, dus mag niet losstaan. Als alternatief kun je een complex maken door `complex(a,b)`.

```
>>> x = 1j
>>> x**2
(-1+0j)
>>> y = complex(2,3)
>>> x + y
(2+4j)
>>> x.real
0.0
>>> x.imag
1.0
>>> abs(3+4j)
5.0
```

Dit laatste geeft de 'modulus' van het complex getal: `abs(x) = sqrt(x.real**2 + y.real**2)`

Scripts

Een script is een programma. Het bevat python-commando's. Het linkervenster van de replit-website draagt de naam `main.py`. Dat is de naam van het bestand, waar jouw code in mag komen. Omdat we online werken, moeten we niet zelf beslissen waar dat bestand bewaard wordt. Willen we de code uitvoeren, dan druk je bovenaan op de groene knop. Stel bv. dat `main.py` volgende tekst bevat:

```
x = 4.0
print ("hallo!")
print (x + 1.0/x)
```

Laat je het programma lopen, dan komt de uitvoer in het zwarte venster rechts.

```
hallo!
4.25
```

Om output naast elkaar te printen, geef je na het eerste stuk expliciet aan dat het einde van de output niet standaard is (standaard = nieuwe lijn klaarzetten voor volgende output), maar een spatie. Zo levert de input

```
x = 4.0
print ("hallo!", end=' ')
print (x + 1.0/x)
```

de output

```
hallo! 4.25
```

Keuzestructuren

Het `if/else`-statement wordt gebruikt om een keuze te maken:

```
if x >= 0:
        print (math.sqrt(x))
else:
        print ("negatief")
```

Let erop dat de if- en else-regel op een dubbele punt eindigen! Bovendien moet je inspringen (liefst met TAB) om aan te geven wat er "in de if" en "in de else" staat. Je kunt het `else` stuk weglaten, en je kunt meerdere commando's inspringen. Bv.

```
if x > 0:
        y = math.sqrt(x)
        print (y)
```

Vergelijkinsoperatoren en combinaties:

<	kleiner dan
<=	kleiner dan of gelijk aan
>	groter dan
>=	groter dan of gelijk aan
==	gelijk aan
!= of <>	verschillend van
and	en
or	of
not	niet

Voorbeeld:

```
if a!=b and (not a>0 or b==2):
        ...
```

Let op het verschil tussen `=` en `==`. Het eerste ("toekenning") wordt gebruikt om een variabele een waarde te geven. Het tweede ("vergelijking") om twee waarden te vergelijken (geeft `True` of `False` als resultaat)

Herhalingsstructuren

Met een herhalings- of lus-structuur geef je aan dat een stuk code meermaals moet uitgevoerd worden. De meest gebruikte zijn de `while`-lus en de `for`-lus.

```
for i in range(10,20):
        print (i,end=' ')
```

Geeft als output:

```
10 11 12 13 14 15 16 17 18 19
```

De "teller" `i` neemt de waarden 10 t.e.m. 19 aan, en voor elke waarde wordt het print-statement uitgevoerd. Zoals bij de `if` moet je indenteren (inspringen).
Merk op dat de eindgrens (20) zelf niet behandeld wordt. Als je dit wil moet je `range(10,21)` schrijven. Het kommaatje achteraan het print-statement zorgt ervoor dat de output op een regel blijft.

Je kunt meerdere regels opnemen in de for-lus door in te springen. De lus stopt op het moment dat je niet meer inspringt:

```
x = 0
for tel in range(0,10):
        print ("hallo")
```

```
        x = x + 10
print x
```

Het laatste commando staat niet ingesprongen ("zit niet in de lus"), en wordt dus *eenmaal* uitgevoerd, *na* de for-lus. In dit geval krijg je tien keer `hallo` op het scherm (onder elkaar), gevolgd door de eindwaarde van x, namelijk 100. Merk op dat de teller `tel` hier niet echt gebruikt wordt, behalve om aan te geven dat je iets 10 keer wil doen.

Alternatief: de `while`-lus.

```
i = 0
while i<10:
        print (i,end=' ')
        i = i+1
```

heeft precies hetzelfde effect als de eerste `for`-lus hierboven. Let op het verschil met de `if`-structuur: bij de `while` wordt het ingesprongen gedeelte (hier `print i,`) *herhaaldelijk* uitgevoerd, *zolang de voorwaarde waar (True) is*.

Let op dat je het laatste commando `i = i+1` niet vergeet; anders blijft `i` op nul staan, en wordt de while-lus nooit afgebroken. Dit noemt men een "oneindige lus". Je moet het programma dan handmatig afsluiten (in de shell bv. met Ctrl-C).

Met de while lus heb je iets meer controle. Je kunt dezelfde vergelijkings- en logische operatoren gebruiken als bij de `if`. Volgend programma berekent de som 1+2+3+4+... en stopt aan 20 of indien de som groter dan 100 wordt. De getallen worden uitgeschreven, en op het einde wordt de som uitgeschreven.

```
a = 1
som = 0
while a<=20 and som<100:
        print (a,end=' ')
        som = som + a
        a = a + 1
print ("som:", som)
```

Voor meer informatie...

leen.brouns@ugent.be
Opleiding Industrieel Ingenieur Informatica
Faculteit Ingenieurswetenschappen en Architectuur
Universiteit Gent