Snelcursus Sage

1. Sage

Het programma **Sage** (in latere versies SageMath genoemd) is een computeralgebrasysteem dat in het jaar 2005 ontwikkeld werd door William Stein, wiskundige en professor aan de universiteit van Washington. Het is een opensourceprogramma dat een waardig tegengewicht wilde brengen voor commerciële wiskundeprogramma's zoals Maple, Mathematica en MATLAB. Pas in 2016 werd de eerste betaalde programmeur aangenomen om Sage verder te ontwikkelen. Momenteel wordt Sage ondersteund door een mix van vrijwilligers en betaalde krachten.

Vele universiteiten die gratis besturingssystemen en gratis software willen propageren, maken gebruik van Sage, zo ook de UGent. Sage is onderlegd in algebra, analyse, combinatoriek, getaltheorie, statistiek, numerieke analyse en grafentheorie. Maar Sage heeft ook enkele onverwachte talenten. Het bevat bijvoorbeeld een automatische sudokusolver.

Het programma Sage maakt gebruik van de sterkste kanten van verschillende andere wiskundige softwarepakketten zoals C++, Lips, Fortran en Python. Vooral Python is bekend als instapprogramma voor beginnende programmeurs.

SageMath (de opvolger van Sage) maakt gebruik HTML (HyperText Markup Language) om teksten op te maken. HTML is sinds lange tijd de standaardtaal op het wereldwijde web. Alle webbrowsers begrijpen het. De formules binnen de HTML-teksten worden gezet in LaTeX. Dit is momenteel de wereldwijde standaard voor wiskundige formules. Als je straks tijdens het programmeren opzoekingen wil doen, dan google je best op SageMath (eventueel ook op Sage of op Python).

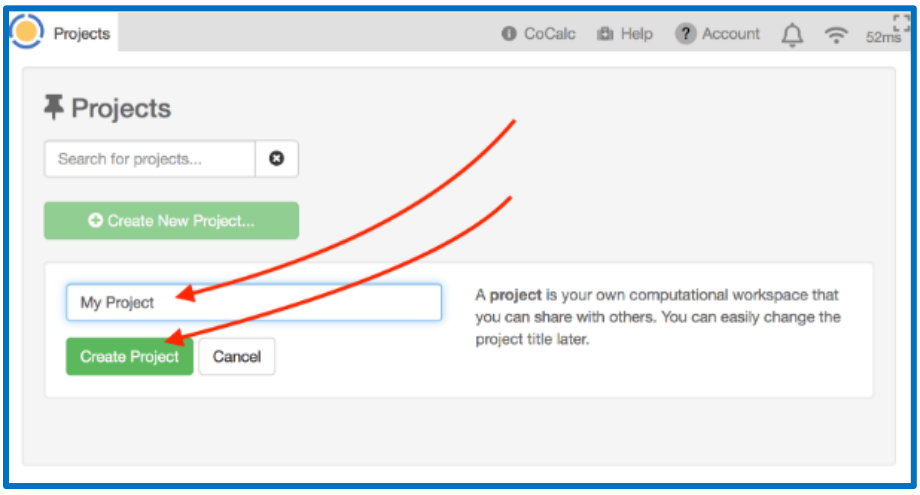
CoCalc is een online virtuele werkruimte waarin je berekeningen en onderzoek kan doen. Je kan er SageWorksheets (*.sagews) maken, die je in de cloud bewaart. Het enige wat je in dit project dus nodig hebt is een webbrowser en een internetverbinding. Geen gedoe met het installeren van software. Terwijl je een programma in CoCalc schrijft, kan je chatten met collaborators. Je hebt tijdens het schrijven van je verslag in Overleaf wellicht gemerkt dat dit handig kan zijn.

2. Een account maken en een programma opstarten

Ga naar de site van CoCalc en maak hier een nieuwe account aan. Vergeet niet je akkoord te geven voor de Terms of Service. Je kan intekenen via een van je mailadressen, bijvoorbeeld het toren.edugo.be-adres. Maar je kan je nieuwe account ook activeren via je Google-account, via

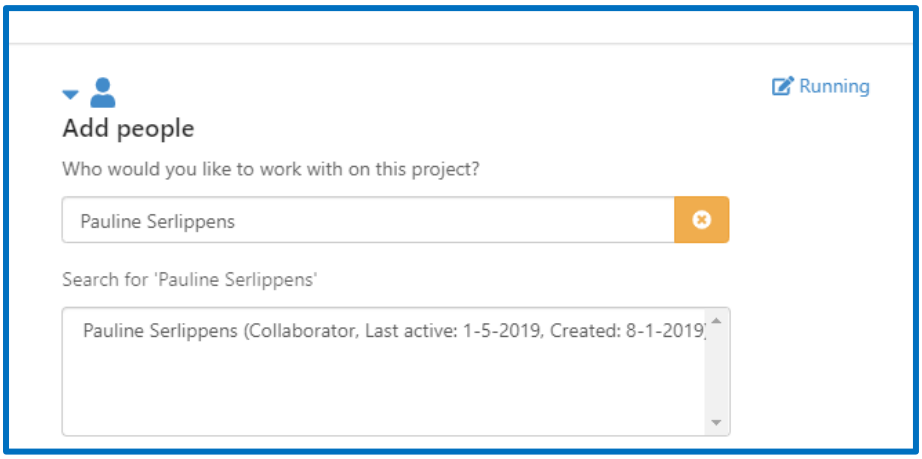
Facebook of via Twitter. Als je dit doet, hoef je geen nieuw wachtwoord meer te kiezen en te onthouden.

Als je ingelogd bent, kan je een nieuw project creëren. Kies als naam 'Corona'. We zullen immers een programma schrijven om voorspellingen te doen over een virusepidemie. Een project is een verzameling van verschillende bestanden, bijvoorbeeld Sage Worksheets, pdf-bestanden, LaTeX-documenten, Jupyter Noteboek-bestanden …



Binnen dit nieuwe project maak je een nieuw bestand aan door op ⊕ Create New Project te drukken. Kies voor Sage Worksheet. Noem het bestand 'Snelcursus'. Nu ben je bijna klaar om te starten met programmeren.

Wil je tot slot nog even je leerkrachten als collaborator aanduiden? Zo kunnen ze je online advies geven. Ga naar het tabblad projects linksboven, klik op het blauwe mannetje (add people) en vul hun email-adres in: chiaradevos@hotmail.com en vandenbroeck_luc@telenet.be .



3. Opdracht

Hieronder zie je een tiental blokjes programmatuur in Sage. Kopieer ze één voor één op een nieuwe regel van je programma.

Telkens wanneer je een blokje hebt ingevoerd, druk je op Run om het te laten lopen. Zolang CoCalc nadenkt over de opdracht, zie je groen horizontaal lijntje knipperen. Dat lijntje staan symbool voor werkende hersenen. Als CoCalc te lang moet nadenken, heb je wellicht een foutje gemaakt in de formulering van je opdracht. Je kan het denkproces dan onderbreken met Stop. Als je fout verbeterd is dan kan je op Restart drukken.

Na elke input in deze tekst volgt een tekstkader. Hierin kan je invullen waarvoor deze programmaregels dienen en wat je van dit stukje programmatuur moet onthouden.

We helpen je kritisch na te denken over de programmaregels door je vooraf een paar vragen te stellen. Aan de hand van deze vragen kan je wat experimenteren.

Deel 1: een centrale titel

Input:

%html   Snelcursus Sage

Vragen:

- Wat is de betekenis van %html?
- Wat betekenen de eerste vier tags tussen <>?

- Wat is de betekenis van de tags met de backslash?
- Waarom gebruiken we aanhalingstekens in CoCalc?

Verklaring:

- Met dit programmablok wordt er een vette titel gezet.
- %html is de start van een html-statement.
- De vier eerste tags tussen <> bevatten de opmaak van de tekst nl kleur, vet, uitlijning en tekstgrootte
- De tags met slash dienen om de opmaak stop te zetten zodat tekst die erop volgt niet op dezelfde manier opgemaakt zou worden.
- Aanhalingstekens laten zien dat het om pre-gedefinieerde woorden gaat.

Deel 2: verklarende tekst in HTML

Input:

%html   Teksten in Sage worden steeds in html (HyperText Markup Language) genoteerd, een taal die geschikt is voor het wereldwijde web en die kan gelezen worden door webbrowsers. Kenmerkend voor html zijn de markeringen (tags) met puntige haken. Sommige markeringen omarmen een tekstblok zoals en waarmee je stukjes tekst vet maakt. Andere markeringen staan alleen, zoals de markering
 waarmee je een regel afbreekt. Let er vooral op dat je in html voortdurend een voorwaartse slash gebruikt.

Vragen:

- Waarvan is
 de afkorting? En ?
- Wat gebeurt er als je weglaat?

Verklaring:

- Met dit programmablokje wordt er een tekstblok toegevoegd aan het bestand. Zo kan je uitleg tussen de berekeningen zetten.
-
 staat voor break, staat voor vette tekst.
- Als je een tag voor de opmaak niet afsluit dan zal overeenstemmende opmaak bewaard blijven.

Deel 3: formules in verklarende tekst opnemen

Input:

%html <font size=4> <font color='darkblue'> Html ondersteunt ook formules in $\LaTeX$. Zo kan je de formule $\Sigma_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$ neerschrijven als je de n eerste kwadraten wil optellen. Let op: bij $\LaTeX$ gebruik je voortdurend de achterwaartse slash.

Vragen:

- Waarom wordt het woord LaTeX als een formule beschouwd?
- Wat gebeurt er als sigma met een kleine letter schrijft?
- Wat gebeurt er als je de tags vergeet te sluiten?

Verklaring:

- In dit programmablok worden teksten gezet waarin wiskundige formules staan.
- De bedoeling is dat het logo van LaTeX verschijnt, niet gewoon "LaTeX". Daarom wordt LaTeX tussen dollartekens geplaatst.
- Het somteken (Griekse hoofdletter sigma) wordt een kleine letter sigma als je sigma met een kleine letter schrijft.
- Als je een tag niet sluit, dan zal de opmaak dezelfde blijven. Als je een dollarteken niet sluit dan krijg je een foutmelding.

Deel 4: enkele eenvoudige programmalijnen

Input:

```
a=4; b=-20; c=21
d=b^2-4*a*c
if d>0:
   print 'de oplossingen zijn', (-b+sqrt(d))/(2*a), 'en', (-b-sqrt(d))/(2*a)
elif d==0:
   print 'de oplossing is', (-b)/(2*a)
else:
   print 'er zijn geen oplossingen'
```

Vragen:

- Wat gebeurt er als je de puntkomma's in de eerste lijn weglaat? Waarom?
- Wat gebeurt er als je in de plaats van puntkomma's op de eerste lijn enters zet en je dus meerdere lijnen van dit commando maakt?
- Wat gebeurt er als je het dubbelpunt na het if-commando weglaat?
- Wat gebeurt er als je de inspringing weglaat de lijn na het if-commando? Zijn ze enkel esthetisch?
- Waarom staan er komma's in het print-commando?
- Waarom staat er twee maal een '=' bij het elif-commando?

Verklaring:

- Dit programma lost vierkantsvergelijkingen op met de discriminantformule.
- Het programma wordt niet uitgevoerd als de puntkomma's weggelaten worden. De uitdrukking $a = 4b = -20c = 21$ betekent immers niets.
- Als je enters plaatst in de plaats van puntkomma's, worden de parameters wel gedefinieerd en zal het programma wel uitgevoerd worden. Statements worden dus gescheiden door harde breaks of door puntkomma's.
- Na de voorwaarde van een if-commando of een elif-commando moet er een dubbelepunt staan.
- Het programma werkt niet als je de insprong bij subroutines niet respecteert.
- In een printinstructie worden de argumenten gescheiden door komma's.
- Een enkel gelijkheidsteken wordt gebruikt om een waarde toe te kennen aan een variabele. Een dubbel gelijkheidsteken wordt gebruikt om een linker- en rechterlid te vergelijken.

Deel 5: werken met lijsten

Input:

```
[i for i in range(10)]
[i for i in range(1,10)]
[i^2 for i in range(10)]
[numerical_approx(random(),digits=4) for i in range(10)]
[p for p in [1,2,..,100] if is_prime(p)]
sum([p for p in [1,2,..,100] if is_prime(p)])
```

Vragen:

- Waarvoor dienen vierkante haken?
- Wat is het eerste getal van range(10)? En het tiende getal van range(10)?
- Wat gebeurt er als je de haakjes na random() weglaat? Kan je dit vergelijken met de input op je zakrekenmachine?
- Wat betekent numerical_approx?

Verklaring:

- In dit programmablok worden alle soorten rijen gedefinieerd. De laatste instructie geeft de som van de 100 eerste priemgetallen.
- De vierkante haken dienen voor de definitie van een rij.
- Een range(n) begint altijd vanaf nul te tellen en gaat tot $n - 1$.
- Het programma wordt niet uitgevoerd. Neen, op je rekenmachine worden er geen haakjes gezet na rand. In Sage moeten er haakjes staan, ook al zijn ze leeg.

Deel 6: grafieken van functies plotten

Input:

```
f(x) = 2^x - 2*cos(x)
g(x)=derivative(f,x)
print 'de functie f(x)=', f(x)
print 'de afgeleide functie g(x)=', g(x)
plot(f, (x, -2, 2), color='green',thickness=3)+plot(g,(x,-2,2), color='magenta',thickness=3)
```

Vragen:

- Wat is het verschil tussen een afgeleide die op je zakrekenmachine berekend wordt en een afgeleide die door Sage berekend wordt?
- Wat gebeurt er als je (x,-2,2) bij beide plots weglaat?
- Wat doet de '+' in de laatste lijn?

Verklaring:

- In dit programmablok worden de grafieken van een functie en de afgeleide functie getekend.
- Met de grafische rekenmachine kan je enkel een afgeleide berekenen in één bepaald punt. Hier leid je een functie af in alle punten. Je berekent m.a.w. de afgeleide functie van een functie.
- In plaats van -2 tot 2 lopen de x-waarden nu van -1 tot 1 (=default).
- Door het +-teken worden beide grafieken in hetzelfde assenstelsel geplot.

Deel 7: een plot maken op basis van een lijst

Input:

```
A=[p for p in [1,2,..,100] if is_prime(p)]
len(A)
B=[(i,A[i]) for i in range(len(A))]
B
u=list_plot(B, plotjoined=True, color='orange',thickness=3)
u
```

Vragen:

- Kan je ook met de lijst A een grafiek plotten? Waarom wel/niet?
- Waarvoor staat de afkorting 'len'?
- Waaraan is len(A) in dit programma gelijk?
- Hoe zorg je ervoor dat er geen 'error' komt als je plotjoined=False zet?
- Waarom staat er geen print voor 'u'?

Verklaring:

- In dit deel wordt een grafiek gemaakt waarbij het i-de priemgetal op de y-as wordt uitgezet tegen het getal i op de x-as.
- Je kan geen grafiek maken van de lijst A. De elementen van lijst A zijn enkel de y-waarden van de punten op de grafiek.
- Len staat voor length. Het getal len(A) is het aantal elementen die de lijst A bevat.
- Als je een puntgrafiek maakt, mag je de punten geen dikte geven. Door "thickness=3" weg te halen vermijd je een foutmelding bij een puntgrafiek.
- Er is een verschil tussen 'plotten' en 'printen'. Een grafiek kan je plotten. Een tekst of een formule kan je printen.

Deel 8: een functiedefinitie met een while-loop

Input:

```
def klokrekenen(a,b):
   while a>b:
      a=a-b
   return(a)
klokrekenen(51,12)
```

Vragen:

- Wat gebeurt er als je return(a) binnen de while-lus zet, dus als je ze verder laat inspringen?
- Wat is dus het verschil tussen print en return?
- Voor welke wiskundige operatie dient de functie 'klokrekenen'?
- Wat gebeurt als bij de instructie klokrekenen(a, b) als a kleiner is dan b?

Verklaring:

- Dit programmablok berekent de rest van een getal a dat gedeeld wordt door een getal b.
- Door de RETURN te laten inspringen ter hoogte van $a = a - b$ wordt de WHILE slechts één keer doorlopen. Na één aftrekking wordt het resultaat al gegeven.
- PRINT betekent dat het resultaat moet gedrukt worden. RETURN betekent dat het resultaat moet doorgegeven worden als het programma ergens wordt opgeroepen.
- Klokrekenen staat voor de wiskunde bewerking 'modulorekenen'. Je berekent de rest van een deling. Dat doe je ook bij het kloklezen. Als iemand zegt dat het 51 uur is dan weet je dat je van dit getal enkele keren 12 uur mag aftrekken. Je berekent de rest bij deling door 12.
- Als a kleiner is dan b dan zal a de rest zijn van de deling van a door b.

Deel 9: een functiedefinitie met een for-loop

Input:

```
def fibo(n):
   F=[1 for i in [0..n]]
   for i in [2..n]:
      F[i]=F[i-2]+F[i-1]
   return(F)
fibo(20)
```

Vragen:

- Wat gebeurt er als je de tweede lijn weglaat? Waarom?
- Waarom kan je de parameter i niet laten lopen in [1..n]?
- Hoe heten de getallen uit fibo(20) officieel?

Verklaring:

- In dit onderdeel bereken je de getallen van Fibonacci: 1, 1, 2, 3, 5, 8, 13, 21 …
- De rij van Fibonacci ligt pas vast als de eerste twee elementen F(0) en F(1) gedefinieerd zijn. Dit gebeurt in de tweede lijn van dit programma. Als de twee kleinste elementen van de rij ontbreken, kan de rij niet opgebouwd worden.
- Als je F(1) zou berekenen met de recursieformule dan lukt dit niet. Er is namelijk slechts één van de voorgangers gekend, nl F(0). F(-1) is immers niet gedefinieerd.
- Vergeet niet dat achter een for een dubbelepunt hoort.

Deel 10: een serie grafieken in één assenstelsel plotten

Input:

```
def kleurigeperioden(n):
   f=sin(x)
   v=plot([])
   for i in [0..n-1]:
      v=v+plot(f, (x, i*2*pi, (i+1)*2*pi), color=hue(random()),thickness=3)
   return(v)
kleurigeperioden(5)
```

Vragen:

- Waarvoor dient de derde lijn? Wat doet deze lijn?
- Wat is de betekenis van hue? En van hue(random())?
- Waarom staat ‘hue(random())’ niet tussen aanhalingstekens zoals wel al steeds het geval was bij kleuren?

- Waarom gebruiken we het interval [0..n-1] en niet [0..n]?

Verklaring:

- Hier wordt een hele reeks grafieken in een assenstelsel geplot.
- In de derde lijn wordt een lege plot v gedefinieerd. Die heb je nodig om er stap voor stap nieuwe plots aan toe te voegen met de operatie +.
- De betekenis van hue is tint en die van random is willekeurig. Er worden dus steeds willekeurige tinten gekozen. Hue is een numerieke kleurwaarde.
- De functie COLOR werkt o.b.v. een numerieke waarde of een pre-gedefinieerde waarde (e.g. “dark-blue”). Om de pre-gedefinieerde waarden numeriek te maken moet er quotes gebruikt worden. Getal-waarden zoals random() hoeven geen qoutes meer.
- Voor n periodes, moet er een lus zijn die loopt van 0 keer 2π tot $n-1$ keer 2π. De parameter i moet dus van 0 tot $n-1$ gaan.

Deel 11: grafieken maken met schuifknoppen

Input:

```
@interact
def paraboolmetschuifknoppen(a=slider(-10,10,1,2),b=slider(-5,5,1,1),c=slider(-5,5,1,-3)):
   f(x)=a*x^2+b*x+c
   v=plot(f,(x,-5,5), color='olive', thickness=3)
   show(v)
```

Vragen:

- Wat doet @interact?
- Wat betekent het woord ‘slider’?
- Waarvoor staan de vier getallen binnen de slider-functie?

Verklaring:

- @Interact zorgt ervoor dat de gebruiker tijdens het lopen van het programma nog een input kan door, bijvoorbeeld door het gebruik van schuifknoppen of van invulvakjes.
- Het woord ‘slider’ betekent ‘schuifknop’.
- Het eerste getal staat voor de minimumwaarde, het tweede voor de maximumwaarde, het derde de stappen waarbij de sliders verspringen en het vierde voor de startwaarde (bij RUN).