Details

This is a take home quiz to test your knowledge of Python in application to the Reactor Design. You are allowed to use any textbooks, look through online resources such as stackoverflow and use the material discussed in class/tutorials. You are strongly encouraged to invest time into this quiz since it will help you catch up with the Python course. You can also use online resources such as https://www.codecademy.com/learn/learn-python to help you grasp the basics of python better. Another good video is https://www.youtube.com/watch?v=ezuRn1VilJE.

Details about the quiz:

Start time: Thursday March 8 3pm

Submission deadline: Monday March 12 6 pm

To submit your quiz finish the exercises in the Jupyter notebook and download it as a PDF and print it. If something goes wrong in opening the Jupyter notebook you may open the PDF (with the quiz) copy paste into a new notebook and then save it. Make sure to include your name and student number.

I strongly encourage you to open your emails with the information about the anaconda python distribution. Syzygy works as well, but you may experience some problems while running it.

Good luck!

Student name: < >

Student number: < >

Problem 1: basics of if/else

Design a function that will calculate a maximum of two numbers:

Call this function `max(a, b)`

Sample output:

```
max(2, 3)
3
max(4, 19)
19
max(3, 3)
3
```

Write the code in the cell below and test your code on the examples above.

Problem 2: basics of loops

Design a function that will calculate factorial of a number.

By definition factorial of a number is multiplication of all numbers that are <= of the given number, e.g. `4! = 1*2*3*4 = 24`

Name the designed function as `factorial(k)`. The sample output will be:

```
factorial(1)
1
factorial(2)
2
factorial(3)
6
factorial(4)
24
```

Write the code in the cell below and test your code on the examples above.

Problem 3: matplotlib and numpy

Lets use the power of Python to tackle CHBE 346 problems.

Calculate and plot the pressure of saturated vapor of ethanol as a function of temperature in the temperature range from $T_{min} = -57^oC$ to $T_{max} = 80^oC$. Create two plots one for pressure in mmHg and the other in kPa.

In case some of you have forgotten (but I doubt you have) here is an inspiration on where to get the equation for the saturated pressure:

https://en.wikipedia.org/wiki/Antoine_equation

To plot you can use the following preamble to import the neccessary python modules:

```
import numpy as np # our matlab-like module
import matplotlib.pyplot as plt # plotting modules
plt.style.use('presentation') # just have in your script for prettier plotting
%matplotlib inline
# if 'presentation' doesn't work use 'seaborn' or 'ggplot'
```

Make sure you assign the xlabel and ylabel

```
plt.xlabel('T [C]')
plt.ylabel('P [mmHg]')
```

In case you want to save what you have plotted you can use this command:

```
plt.savefig('pressure_temp.png')
```

Problem 4: solving ODEs for Batch reactor problems

Zero-th order chemical reaction.

Imagine you have got the simplest possible reaction kinetics for your batch reactor

$-r_A = k > 0$

Your design equation for this reaction in Batch is:

$dC_A/dt = -k$, where

$C_{A0} = 10$ moles/L

$k = 0.1$ moles/(s*L)

1. Calculate and plot: Analytical solution for the given design equation with given parameters C_{A0} and k
2. Calculate and plot: Numerical solution for the same equation using the scipy module `odeint`. Plot curves from 1 and 2 on the same graph.
3. Find time t^* when the concentration of the reacting component is equal to zero, i.e. $C_A(t^*) = 0$

For your code below you can use the following preamble:

```
import matplotlib.pyplot as plt # plotting modules
%matplotlib inline
plt.style.use('presentation') # just have in your script for prettier plotting
# insted of 'presentation' you can use 'ggplot'
import numpy as np # our matlab-like module
from scipy.integrate import odeint # integration of ODEs so you don't have to
write your finite difference yourself
```

Problem 5: CSTR and solving algebraic equations

Given a continuously stirred tank reactor with a volume V = 66 m^3 with a reaction $A \rightarrow B$. The rate of the reaction is:

$-r_A = kC_A^2$ (k=3 L/mol^3/h)

The entering molar flow $F_{A0} = 5mol/$h and the volumetric flow rate is $v_0 = 10L/h$

1. Given that the exit concentration is $C_A = 0.005$ mol/L, how much of A (in percent) has reacted away?
2. If v_0 = 10 L/h what would be the exit concentration C_A?
3. For different values of
   $v_0 = range(1, 21) = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]$ L/h what would be the exit concentration C_A? Plot C_A as a function of v_0.

Just a reminder for a CSTR with zero accumulation rate the design equation looks like this:

$0 = F_{A0} - F_A + V \cdot r_A$, where

$F_A = v_0 \cdot C_A$

To solve the algebraic equation please use the following preamble:

```
from scipy.optimize import fsolve
import numpy as np
```

To do the step 3: you can follow this template:

```
v_0_array = np.arange(1,21)
C_A_array = np.zeros(20) # just allocating the memmory with dummy values 0

for i in range(1,21):
    v0 = v_0_array[i]
    C_A_array[i] = # do the calculation here and put the value C_A
```

Then to plot:

```
import matplotlib.pyplot as plt # plotting modules
%matplotlib inline
plt.style.use('ggplot')
import numpy as np # our matlab-like module

plt.plot(v_0_array, C_A_array, 'go--')
```

Problem 6: Linear regression

1. Fit the data below using $f(x) = A \cdot x \cdot e^{-k \cdot x}$
   a. Plot (xdata, ydata) and (xdata, f(xdata) with the optmized parameters
   b. Print the optimized parameters

2. (Bonus) Calculate the χ^2 parameter of fitting
   $\chi^2 = \Sigma(ydata[i] - f(xdata[i])^2$

You can use this preamble for this exercise:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('presentation') # insted of 'presentation' you can use 'ggplot'
from scipy.optimize import curve_fit # this guy will help us fit everything
```

Data:

```
import numpy as np
xdata = np.array([0.        , 0.02020202, 0.04040404, 0.06060606, 0.08080808,
       0.1010101 , 0.12121212, 0.14141414, 0.16161616, 0.18181818,
       0.2020202 , 0.22222222, 0.24242424, 0.26262626, 0.28282828,
       0.3030303 , 0.32323232, 0.34343434, 0.36363636, 0.38383838,
       0.4040404 , 0.42424242, 0.44444444, 0.46464646, 0.48484848,
       0.50505051, 0.52525253, 0.54545455, 0.56565657, 0.58585859,
       0.60606061, 0.62626263, 0.64646465, 0.66666667, 0.68686869,
       0.70707071, 0.72727273, 0.74747475, 0.76767677, 0.78787879,
       0.80808081, 0.82828283, 0.84848485, 0.86868687, 0.88888889,
       0.90909091, 0.92929293, 0.94949495, 0.96969697, 0.98989899,
       1.01010101, 1.03030303, 1.05050505, 1.07070707, 1.09090909,
       1.11111111, 1.13131313, 1.15151515, 1.17171717, 1.19191919,
       1.21212121, 1.23232323, 1.25252525, 1.27272727, 1.29292929,
       1.31313131, 1.33333333, 1.35353535, 1.37373737, 1.39393939,
       1.41414141, 1.43434343, 1.45454545, 1.47474747, 1.49494949,
       1.51515152, 1.53535354, 1.55555556, 1.57575758, 1.5959596 ,
       1.61616162, 1.63636364, 1.65656566, 1.67676768, 1.6969697 ,
       1.71717172, 1.73737374, 1.75757576, 1.77777778, 1.7979798 ,
       1.81818182, 1.83838384, 1.85858586, 1.87878788, 1.8989899 ,
       1.91919192, 1.93939394, 1.95959596, 1.97979798, 2.        ])

ydata = np.array([ 2.81790690e-03,  5.90294109e-02,  1.09370354e-01,
 1.31612856e-01,
        1.33521472e-01,  1.55179411e-01,  1.59437733e-01,  1.43762661e-01,
        1.35183487e-01,  1.16512815e-01,  1.07702053e-01,  9.67242987e-02,
        1.12202764e-01,  6.90304807e-02,  7.15441262e-02,  6.79904733e-02,
        3.16005654e-02,  4.56444047e-02,  3.32902755e-02,  4.51933972e-02,
        2.97970567e-02,  2.11531961e-02,  1.70962388e-02,  1.53322820e-02,
        1.61206486e-02,  3.31898262e-02,  1.08205741e-02,  1.65291108e-02,
        4.24013381e-04, -3.47150134e-03,  8.98809610e-03,  1.61384528e-02,
```

```
 1.89444656e-02,  1.05952488e-03,  5.15202567e-03,  1.20937619e-02,
-1.53297244e-02, -4.98358443e-03, -3.04000556e-03,  4.59154157e-04,
-1.00965453e-02,  1.08341542e-02,  5.16711834e-03, -2.15908939e-04,
 4.96411518e-04,  3.92671205e-03,  6.95104585e-04,  8.02129061e-03,
 1.74642041e-03, -2.13299969e-03, -1.44225240e-03, -3.91799324e-03,
-8.78295329e-04,  1.04170809e-02, -1.35132175e-02,  2.11129087e-03,
-7.43327553e-03,  1.21894042e-04,  6.57066929e-03,  1.86150924e-03,
 7.46818689e-03,  3.02554354e-03, -2.50852788e-03,  1.71197306e-02,
-3.17197910e-05,  1.31898549e-02,  1.55137128e-03,  1.04708955e-03,
 1.60318797e-02,  2.29432131e-02,  1.61009431e-02, -8.61571685e-03,
-2.36921503e-03,  1.26538983e-03, -1.50510060e-02,  1.22197433e-02,
 1.11696644e-02, -3.66362776e-03,  9.24461107e-03,  1.15448488e-02,
 5.81401176e-04,  1.07067752e-03, -8.02071597e-03,  6.32535856e-03,
-4.05432003e-03, -1.64541547e-02,  2.99370980e-03,  2.04344112e-03,
 2.07861200e-02,  1.25646189e-02, -5.45704279e-04, -5.27434229e-03,
 9.66709541e-03, -8.43414313e-03,  4.08371369e-05, -5.36371330e-03,
-3.27141551e-03,  2.57467424e-03,  8.27639029e-03, -8.23343126e-03])
```