Excercise

- Write down the main topic/title of the last five classes on Python
- For each class, what were the main ideas
  - Outline each class.
  - Put in details
  - What examples were done?
- Do this alone, then with your neighbor.
- Create your own review notes (like these slides).
  - First “recall” then look up to fill in.

Class 7: Python basics, Jupyter cells

Python basics, Jupyter

Code organization, markdown, equations

Functions

Conditionals, units

*Loops

Review, Midterm 2

- Notebook cells
  - Markdown
  - Code
- Shortcut keys
- Python code
  - Variables, acceptable names
  - Comments
  - Indentation
  - Strings, numbers, scientific notation
    - Use 8E5, not 8*10**5
    - Strings: single and double quotes are the same
  - Print
  - Getting help
  - Separating statements in a single line
  - Wrapping to a new line
  - *, /, +, -, **

Class 8: Organization, Markdown, LaTeX equations

Python basics, Jupyter

Code organization, markdown, equations

Functions

Conditionals, units

*Loops

Review, Midterm 2

- Organization
  - Variable = value # units and/or description
  - Add vertical whitespace (blank lines) to separation sections/ideas
  - #---------- to separate sections

```
n  = 2          # kmol
T  = 300        # K
V  = 1.5        # m3
Rg = 8314.46    # J/kmol*K

P  = n*Rg*T/V   # Pa

print(P)
```

- Markdown
  - # Heading, ## Subheading, ### etc.
  - Lists: use * and indented *, or 1. for numbered
  - Links: [Class website](http://ignite.byu.edu)
  - **this is bold**, *this is italic*, ***bold+italic***
  - Format as code: triple back ticks ```print(a + b)```
  - Tables
  - Images: <img src=some_image.png width=300>

Class 8: Organization, Markdown, LaTeX equations

Python basics, Jupyter

Code organization, markdown, equations

Functions

Conditionals, units

*Loops

Review, Midterm 2

- Latex
  - In Markdown cells, equations go between $, or $$
  - Greek symbols: \alpha, \delta, \Delta
  - Subscripts: a_b
  - Superscripts: a^b
  - Wrap in {}: T_{max}
  - Fractions: \frac{numerator}{denominator}
  - Key symbols
    - \partial, \infty, \rightarrow, \nabla, \cdot
    - \le, \ge, \equiv, \approx, \sim
    - \right(, \left), \right[, \left], \right. \left|
    - \sin, \cos, \tan, \exp, \ln, \log_{10}
  - Integral: \int_a^bf(x)dx
  - Sum: \sum_{i=1}^n x_i
  - Text: \mbox{some text}
  - \, \!

Class 9: Functions

Python basics, Jupyter

Code organization, markdown, equations

Functions

Conditionals, units

*Loops

Review, Midterm 2

- Basic syntax and useage →
  - Keywords: def, return, :
  - Function arguments: vars between ()
  - Default arguments
    - Put last
    - Can call with explicit name
  - Docstring between triple quotes
    - help(function_name)
  - return statement
    - Optional
    - Can return more than 1 thing
- Defining vs calling a function
- Function call versus function name
- Pass functions as arguments →
- Variable scope
  - Global, local
  - Names: arguments, vs others

```
def force(m, g=9.81):
    '''
    Get gravitational force.
    m: input: mass (kg)
    g: input: grav. accel (m/s2)
    returns: force (N)
    '''

    return m*g        # Newtons

#-----------------------

m = 25          # kg
F = force(m)    # N
```

```
def integrate(f, a, b):
    m = 0.5*(a+b)
    dx = m-a
    I = dx*0.5*(f(a)+2*f(m)+f(b))
    return I

#-----------------

def f2(x):
    return x**2

#-----------------

Ia = integrate(f2,     1, 2)
Ib = integrate(np.exp, 1, 2)
```

Class 10: Conditionals, Units

Python basics, Jupyter

Code organization, markdown, equations

Functions

Conditionals, units

*Loops

Review, Midterm 2

- Basic syntax and useage →
  - Keywords: if, elif, else, :
  - ==, !=, <, <=, >, >=,
  - and, or, not, true, false

```
if x > 2:
    print("x>2")
elif x > 3:
    print("x>3")
else x > 5:
    print("x>5")
```

```
if x > 2:
    print("x>2")
elif x > 3:
    print("x>3")
```

vs

```
if x > 2:
    print("x>2")
if x > 3:
    print("x>3")
```

- Compound conditional

```
if (x > 3 and x < 6) or x <= 1 :
    print("do something")
```

- Compact if statement

```
x = np.sqrt(y) if y>0 else np.sqrt(-y)
```

Class 10: Conditionals, Units

Python basics, Jupyter

Code organization, markdown, equations

Functions

Conditionals, units

*Loops

Review, Midterm 2

- Units
  - Convert units at the beginning to consistent system: SI or English
  - Do calculations
    - Avoid doing unit conversions in intermediate calculations
  - Convert units for reporting results if needed
  - Use absolute temperature scales
    - Unless a particular correlation requires oF or ^{o}C, as in the above, where A, B, C are given assuming T in ^{o}C, for example.

$$P_i^{sat} = A + \frac{B}{T + C}$$

  - **Unit conversions are straightforward, but are a common source of error. Be careful.**
  - **Always label all variables and expressions with units in a comment statement to the right.**

Class 10: Conditionals, Units

- Pint

Python basics, Jupyter

Code organization, markdown, equations

Functions

Conditionals, units

*Loops

Review, Midterm 2

```
import pint                 # import the library
u = pint.UnitRegistry()     # get the units from the UnitRegistry

#--------------------

L = 5 * u.m     # meters
t = 2 * u.s     # seconds

#--------------------

L.magnitude      # number part of L without the unit part
L.units          # unit part of L without the number part

#--------------------

v1 = L/t              # vel in m/s: units automatically determined
v2 = v1.to(u.ft/u.s) # function "to":  v2 (=) ft/s;  v1 is still m/s
v1.ito(u.ft/u.s)      # function "ito": v1 is now in ft/s

#--------------------

M  = 29      * u.kg/u.kmol
P  = 1       * u.atm
T  = 600     * u.K
Rg = 62.363 * u.L*u.torr/u.mol/u.K

ϱ = M*P/Rg/T

print(ϱ)                      # some complex density unit reported
print(ϱ.to_base_units())      # function "to_base_units"
print(ϱ.to(u.lb/u.ft**3))     # some other desired units
```

Class 11: Loops

Python basics, Jupyter

Code organization, markdown, equations

Functions

Conditionals, units

*Loops

Review, Midterm 2

```
def f(x):  return x**2-3
def fp(x): return 2*x

x = 1
for i in range(100):
    xnew = x - f(x)/fp(x)
    if np.abs((xnew-x)/xnew) < 1E-6:
        break
    x = xnew

print(f"  x = {x:.5f}")
```

```
x = 1.73205
```

- Basic syntax

```
for i in range(10):
    print(i)
```

  - i can be any variable
    - i used to create a loop, but doesn't have to be used inside the loop.
  - range(stop), range(start, stop), or range(start, stop, step)
    - Default starts at 0: values range from 0 to stop-1
  - Define variables inside or out:

```
n = 1
R = 8314
T = 300

for V in range(2,21):
    P = n*R*T/V
    print(P)
```

  - Nested Loops:

```
for i in range(3):
    print(f"i={i}")
    for j in range(4):
        print(f"  j={j}")
```

  - Reset variables:
    - Sum from 1 to 100

```
s = 0
for i in range(1,101):
    s = s+i
print(s)
```

  - Break early
    - Newton's method example for f(x) = x2-3, find x for f(x) = 0

Class 12: Arrays

*Arrays
Variables, scope, tuples, dictionaries
Ploting, numpy file i/o
Modules and classes

- Python lists, numpy array
  - import numpy as np
- Array creation
  - array, arange, linspace, zeros, ones, empty
  - 2D array: np.array([[1,2,3], [4,5,6]])
    - Array of arrays
- Access elements: a[5], a[i], a[i+7]
  - A[-1] = last element; a[-2] = second to last, etc.
- Array slices: a[2:6] --> elements 2, 3, 4, 5
  - a[:], a[3:], a[:-2], a[3:10:2], a[::-1]
- np.size(a), np.shape(a), len(a)
- Solve Ax = b; x=np.linalg.solve(A,b)