Lecture 6 and 7

Python programming:

- Functions
- Modules in Python: Math module
- Coding error propagation formulae

Functions: Python textbook (Liang), chapter 6

https://docs.python.org/3/tutorial/datastructures.html
https://docs.python.org/3/library/math.html

Lecture 6

L6Ex1 (Lecture 5 on Data Analysis)

A bird flies a distance d = 120 ± 3 m during a time t = 20.0 ± 1.2 s.
The average speed of the bird is v = d/t = 6 m/s. What is the uncertainty of v?

$d = 120.0 \pm 3.0$ m
$t = 20.0 \pm 1.2$ s

$\boxed{v = d/t}$

derive formula for uncertainty on v (error propagation)

$$\sigma_v^2 = \left(\frac{\partial v}{\partial d}\right)^2 \cdot \sigma_d^2 + \left(\frac{\partial v}{\partial t}\right)^2 \cdot \sigma_t^2$$

$\frac{\partial v}{\partial t} = -\frac{d}{t^2}$ (treat d as a "constant", when deriving partial derivative with respect to t)

$\frac{\partial v}{\partial d} = \frac{1}{t}$ (treat t as a "constant")

$$\sigma_v^2 = \frac{\sigma_d^2}{t^2} + \frac{d^2}{t^4} \cdot \sigma_t^2$$

check if units make sense

$\left[\frac{\sigma_d^2}{t^2}\right]$ should be in $\left[\frac{m^2}{s^2}\right]$ (since uncertainty on v^2 is $\frac{m^2}{s^2}$)

$\left[\frac{\sigma_d^2}{t^2}\right] = \left[\frac{m^2}{s^2}\right]$

$\left[\frac{\sigma_t^2}{t^4} \sigma_t^2\right]$ Should also be in $\left[\frac{m^2}{s^2}\right]$

$\left[\frac{\sigma_t^2}{t^4} d^2\right] = \left[\frac{s^2}{s^4} \cdot m^2\right] = \left[\frac{m^2}{s^2}\right]$

$$\boxed{\sigma_v = \sqrt{\frac{\sigma_d^2}{t^2} + \frac{\sigma_t^2}{t^4} \cdot d^2}}$$

Use error propagation formula:

$$\sigma_f^2 = \left(\frac{\partial f}{\partial x_1}\right)^2 \sigma_{x1}^2 + \left(\frac{\partial f}{\partial x_2}\right)^2 \sigma_{x2}^2 + \ldots\ldots + \left(\frac{\partial f}{\partial x_n}\right)^2 \sigma_{xn}^2$$

And write a python code that gives Numerical values for v and sigma_v

L6Ex1 solution-python code

```
#Write a corresponding program using uncertainty formula
#derived on paper.

#declaration of variables
d = 120.      # in meters
sigma_d = 3.  # in meters
t = 20.0      # in sec
sigma_t = 1.2 # in sec

v=d/t

# uncertainty on v squared
sigma_v2 = (sigma_d/t)**2 + (sigma_t*d/t/t)**2
sigma_v = sigma_v2**0.5


#print the results
print(v, sigma_v)    # we will learn later how to round them in python
```

L6Ex2:

A bird #1 flies a distance of 120 $\pm$ 1 m during a time = 20.5 $\pm$ 0.1 s.
A bird #2 flies a distance of 240 $\pm$ 1 m during a time = 10.5 $\pm$ 0.1 s.

Calculate the speed of bird #1 and bird #2 and their uncertainties.
Modify L6Ex1 to include these 2 measurements.

Solution A: Bad programming style (python code)

```
#declaration of valiables
#bird1
d1 = 120.        # in meters
sigma_d1 = 1.  # in meters
t1 = 20.5        # in sec
sigma_t1 = 0.1 # in sec

#bird2
d2 = 240.        # in meters
sigma_d2 = 1.  # in meters
t2 = 10.5        # in sec
sigma_t2 = 0.1 # in sec

v1=d1/t1
v2=d2/t2

# uncertainty on v squared
sigma_v12 = (sigma_d1/t1)**2 + (sigma_t1*d1/t1/t1)**2
sigma_v1 = sigma_v12**0.5

sigma_v22 = (sigma_d2/t2)**2 + (sigma_t2*d2/t2/t2)**2
sigma_v2 = sigma_v22**0.5

#print the results
print(“Bird1: “, v1, sigma_v1)
print(“Bird2: “, v2, sigma_v2)
```

Why is it bad? Imagine having to calculate speeds and uncertanties for 1000 birds …
Codes should be compact, and formulae should be coded <u>once</u>.

Functions

Functions: a collection of commands grouped together that performs a given task.

Functions have arguments (data) do something with the data , and returns a result (non-value or value(s))

Functions can be used many times, and even called by other functions.

Defining a function in a python code:

A function definition consists of the function's name, parameters, body, and return value(s)

```
def functionName(list of parameters separated by commas):
    #function body
    return list of return variables separated by commas
```

All functions should be defined at the beginning/top of your program.

Defining a function (returning value) and invoking it with parameters

#Defining a function

Function's parameters
("LOCAL" VARIABLES: not known to python outside function)

Function name

Function header

```
def sum( n1, n2):
    result = 0
    for i in range(n1, n2+1):
        result += i
    return result
```

Function body

Single return value

Space needed here (indent)

#And invoking it with 2 parameters

```
s1=sum(1,14)     # return value of sum function is assigned
to variable s1
s2=sum(20,53)
s3=sum(100,133)
a=3
s4=sum(a,10)
```

parameters
(can be numerical values or declared/initialized variables)

Functions : Structure of your code

Functions: code that performs a given task. Functions have arguments (data) do something with the data , and returns a result.
Functions can be used many times, and even called by other functions.

Example of a **value returning function**)

```
def function1(x_local):
    #code to do something with x_local
    y_local = x_local +10
    return y_local
def function2(a_local,b_local):
    #code to do something with a_local,b_local
    c_local = a_local + b_local
    return c_local
```

Local variables/names (used only locally, inside a function)

```
#main code
x1_global= 5.
y1_global=function1(x1_global)
x2_global = 6+7
X3_global = 9/7
y2_global=function2(x2_global, x3_global)
```

Functions are typically defined at the top of a program.
They must be defined before they are “called”.

L6Ex3 (illustrates python program structure)

```
#your functions are defined here
def getSquareRoot(x):     # one argument: x
    y = x**0.5
    return y
#main code
x1=49.
x2=getSquareRoot(x1)  # calling getSquareRoot function
                      # with x1 is an argument
                      # x2 is a return value (results)
print (x,y)  ## ? Does it work?
print(x1,x2)
```

Variables x,y are Local variables/names

Math (not a code) :

f(x)=x**0.5   a function, defined for all x. This is NOT a python command.

f(49)=49**0.5 f evaluated for a specific value of x

Corresponding code:

```
def f(x):
    return x**0.5
print ( f(49) )
```

Value returning functions

L6Ex4: define a function that calculates an average of 3 numbers

```
def average(n1,n2,n3):
    print (n1,n2,n3)
    return (n1+n2+n3)/3.0

#main program
result = average (10, 12, 19)
print (result)
print (n1,n2,n3)
```

#n1,n2,n3 are local variables,
#not known outside the function

L6Ex5 Alternative:

```
def average(n1,n2,n3):
    avg = (n1+n2+n3)/3.0
    print (n1,n2,n3,avg)
    return avg
#main program
result = average (10, 12, 19)
print (result)
print (n1,n2,n3,avg)
```

#n1,n2,n3,avg are local variables,
#not known outside the function

(More than one) Value returning functions

```
L6Ex5:

def getMinAndMax(my_list):
        x=max(my_list)
        y=min(my_list)
        print (x,y)
        return x,y

#main code
my_list = [10, 21, 33, 9, -10]
getMinAndMax(my_list)
my_list_min, my_list_max = getMinAndMax(my_list)
print (my_list_min, my_list_max)
```

Trace this code

Functions
Non-Value returning functions

```
#defining a function called display_message,
#which takes no arguments () and returns no arguments
def display_message():
    #code what we want this function to do
    print(‘Here is my message’)
```

list of arguments

name of a function

Indicates definition of a user function in python

Functions
Non-Value returning functions

```
#defining a function called display_message,
#which takes no arguments () and returns no arguments
def display_message():
    #code what we want this function to do
    print('Here is my message')


#main program
#calling function named display_message()
display_message()
```

Lecture 7

Homework 3:
Do all Lecture 7 exercises: L7Ex1, …, up to L7Ex10
(10 separate python scripts OR 1 script that combines all 10 exercises)

DUE: Sept 24 (Tuesday)

Functions

Reminder

Functions: a collection of commands grouped together that performs a given task.

Functions have arguments (data) do something with the data , and returns a result (non-value or value(s))

Functions can be used many times, and even called by other functions.

Defining a function in a python code:

A function definition consists of the function's name, parameters, body, and return value(s)

```
def functionName(list of parameters separated by commas):
    #function body
    return list of return variables separated by commas
```

Example: write a python code that sums numbers from a) 1 to 14, b) from 20 to 53 and c) from 100 to 133

```
sum = 0
for i in range(1,15):
    sum += 1
print('Sum from 1 to 14 is',sum)
```

```
sum = 0
for i in range(20,54):
    sum += 1
print('Sum from 20 to 53 is',sum)
```

```
sum = 0
for i in range(100,134):
    sum += 1
print('Sum from 100 to 133 is',sum)
```

3 groups of very similar code

Bad programming style

Defining a function (returning value) and invoking it wit Reminder ers

#Defining a function

Function's parameters

Function name

Function header

```
def sum( n1, n2):
    result = 0
    for i in range(n1, n2+1):
        print (i, result)
        result += i
        print (i, result)
    return result
```

Function body (local variables)

Single return value

#And invoking it with 2 parameters

```
s1=sum(1,14)
s2=sum(20,53)
s3=sum(100,133)
```

actual parameters (can be numerical values or variables)

Reminder

<u>Example:</u> write a python code that sums numbers from a) 1 to 14, b) from 20 to 53 and c) from 100 to 133

```
def sum(n1, n2):
    result = 0
    for i in range(n1, n2+1):
        print (i, result)
        result += i
        print (i, result)
        return result
```

These lines define a function named **sum,** with 2 arguments (parameters) n1 and n2

```
print('Sum from 1 to 14 is', sum(1,14))
print('Sum from 20 to 53 is',sum(20,53))
print('Sum from 100 to 133 is',sum(100,133))
```

These lines define the **main** function that invokes ("calls") sum(1,14) to compute the sum from 1 to 14, sum(20,53) to compute the sum from 20 to 53, sum(100,133) to compute the sum from 100 to 133,

Trace this code (done in class on whiteboard)

The **module based services model**
- the core idea behind code architecture in Python.

- Every file of Python code whose name ends in a .py extension is called a **module.**
- Other files can access the items defined by a module by importing that module.

e.g.
```
>>> import math
```

Import operations load another file, and grant access to the file's contents.

Contents of a module are available to the outside world through its **attributes**.
(module is a package of names, "a namespace", and the names within that package are called attributes: variable names that are attached to a specific object)
Imports must find files, compile and run the code.

Larger programs take form of multiple module files, which import tools from other files.
One of the modules is designated as the main file. Modules serve the role of libraries of tools.

Math.py: Mathematical module

- math.py (this module is always available)
- In python script: **import math**

List of available functions can be found:
https://docs.python.org/3/library/math.html

Check this out

In math module the argument of trigonometric functions (x) is always in radians.

L7Ex1:

```
import math
x1=30
y1=math.radians(x1)  ## this shows how to use math module
```

if x=30 is given in degrees, one must first convert to radians

```
print(math.sin(x1) )
y1=math.radians(x1).
print (math.sin(y1))
print (math.exp(2))
x1=2; y1=3
print (math.pow(x1,y1))
print (math.sqrt(2) )
print (math.pi)
print (math.e)
```

Constants

Math.py: Mathematical module

- math.py (this module is always available)
- In python script: AT THE VERY TOP OF YOUR PROGRAM
  3 alternative ways you can import modules or specific functions:
  - import math OR
  - from math import * OR
  - from math import sin,radians *(or/and other functions you need)*

L7Ex2

```
from math import sin, radians
x1=30
print(sin(x1) )

y1=radians(x1).
print (sin(y1))
```

for this method you do not need to type math.radians, just radians

Rounding in python code

The function **round(x,n)** takes 2 arguments: x number to be rounded and **n digits from the decimal point**
and returns one value, which is **x** rounded to **n** digits from the decimal point.

Executing in python (either in python script or in interactive python session):

round(80.23456, 2) # this is a python command

will return: 80.23

Example of rounding numbers in python code (with explanation):

```
x1= 980.2675665
sigma_x1 = 0.035678
precision = 3 # based on the value of sigma_x1 you
              # need to figure out how to round to 2 significant digits
              # and decide that 2 significant digits corresponds to 3 decimal places
              # thus precision = 3 (not 2!) in the round method.
rounded_sigma_x1 = round(sigma_x1, precision)
rounded_x1 =  round(x1, precision)
print (rounded_x1, rounded_sigma_x1)
```

User defined value returning function

L7Ex3

Write a code that calculates sin(x)+2cos(x) for x of 10 degrees and of 20 degrees, in a user defined function with argument x.

Note: in your code , follow the structure:

1) First import all needed module(s)
2) Then define all your functions (using local variables)
3) Then write the main program

User defined value returning function

```
L7Ex3 Write a code that calculates sin(x)+2cos(x) for x of 10
degrees and of 20 degrees, in a user defined function with argument
x.

#first import all external modules
import math

#then define your function
# arg1 is an argument of user defined function named
# my_function_name
def my_function_name(arg1):
    arg1_rad = math.radians(arg1)
    f = math.sin(arg1) + 2*math.cos(arg1)
    return f

#main code
print(my_function_name(10.))
print(my_function_name(20.))
# or alternatively:
x1=10.
y1=my_function_name(x1)
print(y1)
x2=20.
y2=my_function_name(x2)
print(y2)
```

Local variables/names (used only locally, inside a function): arg1, arg1_rad, f

L7Ex4:

A = 2 is a constant .

Write a code to calculate values of a function f $f(x) = A\sqrt{x}$
for x=10.56 in 2 ways: with a user-defined function and without

L7Ex5:

A = 2 is a constant .

Write a code to calculate values of a function f $f(x) = A\sqrt{x}$
for x=10.56 in 2 ways: with a user-defined function and without.
Add uncertainty calculation, assuming σ_x / x = 0.10.

L7Ex5:

Use the math module functions such as exp, pow, sqrt to define functions f given below **and their uncertainties sigma_f** (using error propagation method). Formulae for df should be derived on paper, and their final form should be coded in python.

For a given function f, calculate its uncertainty σ_f using known : $x \pm \sigma_x,\ y \pm \sigma_y, z \pm \sigma_z$

a) $f(x) = A\sqrt{x}$

b) $f(y) = Be^{2Cy^2}$

e) f(x)=2sin(x)+1

c) $f(x,y) = \dfrac{x-y}{x+y}$

d) $f(x,y,z) = \dfrac{1}{\sqrt{2}}(x+y+z)$

A,B and C are constants. Define A=2.,B=1.,C=0.5 in the main program, and pass them as arguments to user defined functions. Assume x to be 10.56, 30, 45, σ_x /x = 0.10,
Y to be 0.0987, 0.045, 0.31, σ_y /y = 0.25,
z=3.2*10^3 , 54000, 65.789, σ_z /z = 0.05.

L7Ex6

Textbook problem 3.14 (Lecture 5)

3.14) $t = (3.0 \pm 0.5)\,s$

$d = \frac{1}{2} g t^2$

$g = 9.80 \frac{m}{s^2}$, $\sigma_g = 0$

$d = ?$

$\sigma_d = ?$

$$d = \frac{1}{2} \cdot 9.80 \frac{m}{s^2} \cdot (3)^2 s^2 = 44.1\ m$$

$$\sigma_d^2 = \left(\frac{\partial d}{\partial t} \cdot \sigma_t\right)^2 + \underbrace{\left(\frac{\partial d}{\partial g} \cdot \sigma_g\right)^2}_{=0} = \left(\frac{\partial d}{\partial t} \sigma_t\right)^2 = \left(g \cdot t \cdot \sigma_t\right)^2$$

$$\sigma_d = g t \cdot \sigma_t$$

$$\sigma_d = 9.80 \frac{m}{s^2} \cdot 3\,s \cdot 0.5\,s = 14.7\ m$$

Rounding (2 significant digits) : $\sigma_d \approx 15\,m$

$\Downarrow$

$d \approx 44\,m$

Result: $d \pm \sigma_d = (44 \pm 15)\,m$

very large uncertainty $\frac{\sigma_d}{d} = 0.34$

(34 %)

Write a program that calculates a numerical value of d and its uncertainty. Print round the results using <u>round</u> function

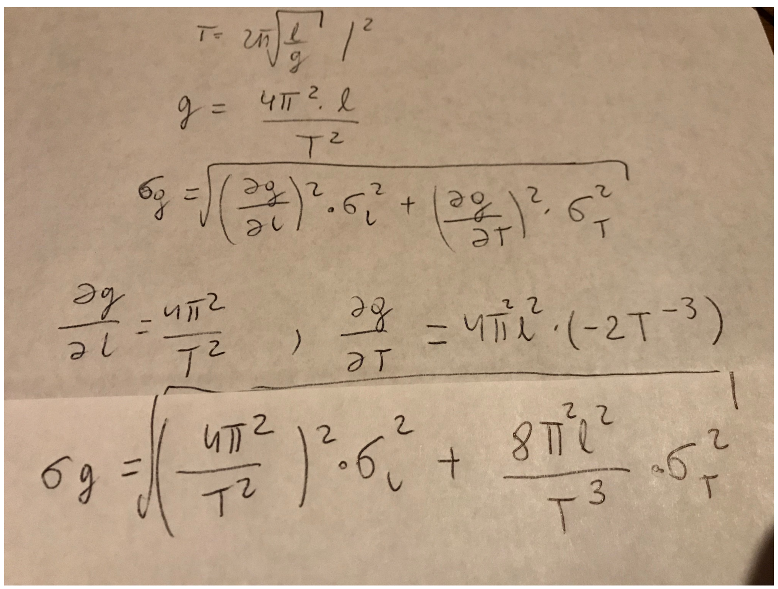
Textbook example Sect. 3.9 (Lecture 5)

L7Ex7

Pendulum experiment.
Find g and its uncertainty using error propagation (2 variables: l and T)

$$l = 92.95 \pm 0.15 \text{ [cm]}$$
$$T = 1.936 \pm 0.004\ [s]$$

$$T = 2\pi\sqrt{\frac{l}{g}}$$



<u>Practice:</u> Write a program that calculates g and its uncertainty for the measured l and T. Print round the results using <u>round</u> function

Practice: value returning functions

L7Ex8: in python define a function that calculates an average of numbers from a given list. Call this function in the main part of your code for the following list of numbers: `10, 12, 19. Code trace it.`

```
def average2(my_list):
    sum=0
    count=0
    for element in my_list:
        sum+=element
        count+=1
    n=count
    return sum/n

x= [10., 12., 19., 20., 56., 78.]
result = average2(x)
print (result)
```

(More than one) Value returning functions

L7Ex9:
for a given list of numbers:10, 21, 33, 9, -10
write a function that takes this list as an argument and returns two values from that list: minimal and maximal. Code trace it.

```
def getMinAndMax(my_list):
    x=max(my_list)
    y=min(my_list)
    #min and max are standard python functions,
    #that return minimal and maximal values from the input list
    print (x,y)
    return x,y

my_list = [10, 21, 33, 9, -10]
getMinAndMax(my_list) #passing the list as an argument
my_list_min, my_list_max = getMinAndMax(my_list)
print (my_list_min, my_list_max)
```

Practice@home: value returning functions

<u>L7Ex10</u> Consider the following height measurements results for a selected group of 35 people:

168.4,165.7,166.8,170.1,169.8,169.2,168.8,
166.2,169.0,160.5,168.8,169.1,168.1,168.0,
166.7,168.8,169.9,170.9,169.6,168.2,167.4,
167.6,167.6,167.8,167.9,168.2,169.0,169.8,
170.6,170.8,169.3,167.6,166.2,163.6,163.8

Write a python code (with 2 functions) to calculate (<u>and print</u>) :
1) the range of the height values (max and min),
2) the mean height, its uncertainty (correctly round the result), and the value of standard deviation.
What do these quantities (from 1 and 2) tell us about this group of people ?
(Give an answer in a print statement)

<u>Helpful formulae:</u>

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad s^2 = \frac{1}{n-1}\sum\left(x_i - \bar{x}\right)^2 \qquad u^2 = \frac{s^2}{n}$$

Homework 3:

Do all Lecture 7 exercises: L7Ex1, …, up to L7Ex10
(10 separate python scripts OR 1 script that combines all 10 exercises)

DUE: Sept 24 (Tuesday)