Lecture 2:

- Define a function in Python:

  EX: ← function name (you set by yourself. ! Don't set to exist names)

  def gaussian(x, mu, sig):

  indicate you defining a function

  2.0 instead of 2 because float gives float

  return exp(-power(x-mu, 2.0)/(2.0 * power(sig, 2.0)))

  ↑ This indicate what after this is the output when function is called.

  Build-in function (python's ~~does~~ function).

- note that if you run the code, there is nothing shown in Shell. unless you 'use' the function, either in Shell or Script.

  Ex: ← set value y to the return value.

  y = gaussian(2, 10, 5)

  print y → let's assume μ=10, σ=5

- To plot a gaussian curve:

  Ex:

  y=[ ]

  set range of x

  x = range(0,10)

  ~~for i~~
  for i in range(0,10):   for-loop to calculate each y-value for x.

  y = y+[gaussian(i, 10, 5)]

  plot(x,y)

  ~~Import matplot~~

  from matplotlib import plot

  this plot points and connect them

Lecture 3:

- List; list is a python 'thing'. we could define it by "[ ]", note that 1st element of the list index as "0".

  Ex:

  A = [1, 2, 3, 4, 5]

  ~~If~~
  print A[0] ⟶ this gives 1.

  A.append(24) ⟶ this add one move to end of list.

  print A[0:2] ⟶ this gives a list: [1,2]

- Array; This is only defined in numpy. OR "array is list in numpy" The reason is to do this it enables many array operators in numpy.

  Ex: ∅

  A = array([1, 2, 3, 4, 5]) → we defie an array using a list.

  Note: Be ~~exten~~erely careful when doing "b=a" when a is a list. You probably means

Lecture 4:
- for-loop: As in word, you are trying to look each element in a list. You do not need to predefine the label (usually i).

Ex:
```
for i in range(5):
    print i.
```
range(5) is a list: [0, 1, 2, 3, 4]

You do not to define i, system knows i=1, then i=2 ....

Ex 2:
p = 0 → p is like empty slot that does not reset for each loop.
```
for i in range(5):
    p = p+i
```
→ so, p keeps grow.

- while-loop: while loop run (excute) the codes below it iff and only if the condition meet, then, if condition not meet, it moves on.

Ex:
```
x = 6
while x > 0:
    x = x+1
    print x
```
This code will run forever

Because x=6, then, while x>0 is true, the code below makes x=7. Then, it goes back check if x>0 again.

Lecture 5:
- Data:
Ex:
```
import numpy as np
thedata = np.loadtxt('filename.txt')
```
← in order to use command

This product a list called thedata.

↑ variable name   ↑ name of file for the data.

• mean (average), usually denote as μ or $\bar{x}$,

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$, x_i is the value for one instance.

Ex:
mean = sum(thedata) / len(thedata)
          ↑   ↑
      sum over all data   divide by length of data
                          a.k.a N.

• variance, denote by σ^2 (pronounce 'sigma')

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2$$
                              ↑ average (mean)

• standard deviation is square root of variance.
$$\sigma = \sqrt{\sigma^2}$$

- Gaussian (Normal):
  - formula:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\bar{x})^2}{2\sigma^2}} \quad ; \quad \bar{x}, \sigma \text{ are defined by the given data.}$$

$\bar{x}, \sigma$ are **not** unknowns.

~~This is NOT the probability given x !!!~~

This is called probability amplitude.

Second thought, this could mean probability at exactly that point...

(Think about it, when you say $p(x=4)$, do you mean x is exactly 4? what about 4.00001? So, probability is defined as a range, like $4.0 < x < 4.1$)

Ex:

Refer to Lecture 1 review.

```
def gaussian (x, mu, sig):
    return exp(- power (x-mu, 2.0) / (2.0 * power( sig, 2.0)))
```

○ Probability: The probability is the area we define the x.
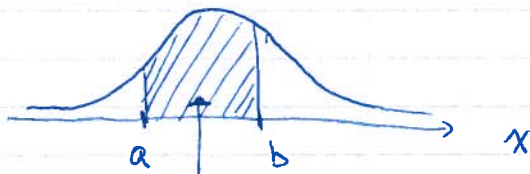
Probability of $a < x < b$.   number, not a variable

so, $prob(a < x < b) = \int_a^b dx \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\bar{x})^2}{2\sigma^2}}$

↑ integrate over x=a to b.

$$= -\frac{1}{2} erf\left(\frac{a}{\sigma}\right) + \frac{1}{2} erf\left(\frac{b}{\sigma}\right) \Longleftrightarrow \frac{1}{2}\left(erf(-a) + erf(b)\right)$$

Probability of get a between 0 to a.   prob of $0 < x < b$.

- $erf\left(\frac{a}{\sqrt{2}\sigma}\right)$: probability get $-a < x < a$.

- $erfc\left(\frac{a}{\sqrt{2}\sigma}\right) = 1 - erf\left(\frac{a}{\sqrt{2}\sigma}\right)$: prob get $-\infty \lesssim x \lesssim -a$ plus $\infty > x > a$

- $\frac{1}{2}erf\left(\frac{a}{\sqrt{2}\sigma}\right)$: prob getting $0 < x < a$

- $\frac{1}{2}erfc\left(\frac{a}{\sqrt{2}\sigma}\right)$: prob getting $\infty > x > a$

sigma ↙

- $\frac{1}{2}erfc\left(\frac{a}{\sqrt{2}}\right)$: prob get $\infty > x > a\sigma$

- $erfc\left(\frac{a}{\sqrt{2}}\right)$: prob get $\infty > x > a\sigma$, plus $-\infty < x < -a\sigma$

↪ This is called a-significant.

lecture 5 (part) - (cont):

Ex: (not python code)

$\frac{1}{2} erf\left(\frac{c}{\sqrt{2}\sigma}\right)$

$\frac{1}{2} erfc\left(\frac{a}{\sqrt{2}\sigma}\right)$

$\frac{1}{2} erfc\left(\frac{b}{\sqrt{2}}\right)$

or b - significance.
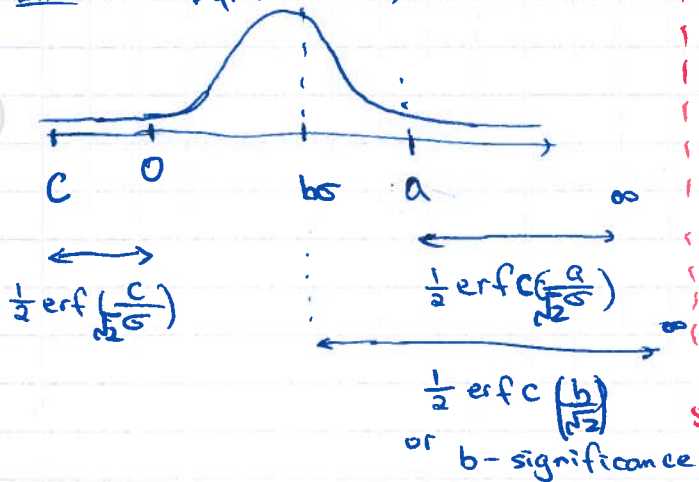
- Model fitting: we when we have to calculate the error:

$$\chi^2 := \sum_i \frac{(y_i - m_i)^2}{2\sigma_i^2} \; ; \; y_i \text{ the actual value. } m_i \text{ the predict value.}$$

↖ variance

$$\chi^2 = \frac{\text{different between predict and real}}{\text{over rescale the number}}$$

because large diff in large variance model is not a big deal.

• So, overall, we minimize the χ^2 value to better fit the data.

Lecture 7:

- $\binom{n}{k}$: This means n choose k. This gives number of ways to draw k items from total of n items.

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} \; ; \; 3! = 3 \times 2 \times 1.$$

- Binomial Distribution:

$$p(k) = \binom{n}{k} p^k (1-p)^{n-k} \; ;$$

→ set this either from a given data or knowledge.
P_e is the probability event A happens.
R_n is the total events number.

Ex: (not a python)

$$p(k) = \binom{10}{k} (0.4)^k (1-0.4)^{10-k} \rightarrow$$

let's say from 10 coin flips what are probs get exactly k heads, when this coin has 40% head chance.

↓ how many combinations of exactly k heads

↓ for 1 combination, the probability.
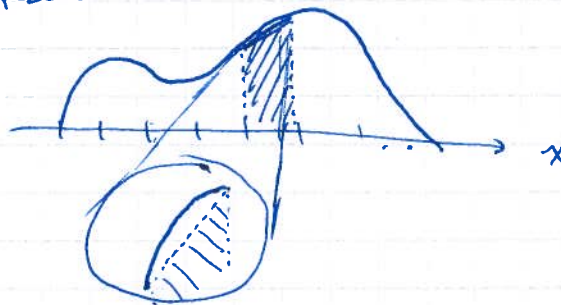
Lecture #9

- Riemann Sum: use midpoint to construct a rectangle.
  Trapezoid Rule: use slopes to construct a Tpzoid.

  Riemann Sum   Trapezoid Rule:

Ex:
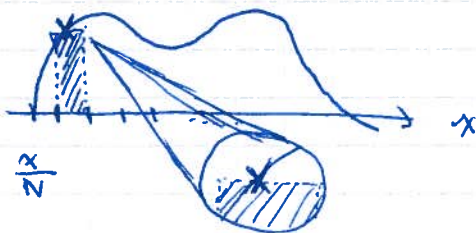
```
x = range (0, 20, 0.01)  →    Set Δx = 0.01
sum = 0                        x is x-axis.
for a in range to x :
    midpoint = a + 0.05          Rie Sum use
    y = f (midpoint)             the midpoint
                                 So, we + half a interval.
    sum += 0.01 * y

print sum                        height of the function
```

Area calculate.   next
                    a:(a+1)
use   ↓
$$A = 0.01 * \frac{f(a) + f(a+0.01)}{2},$$
for Trapezoid Rule.

number for
integrate f over range 0 to 20.

aka $\int_0^{20} f(x) \, dx$

area is this   } y(midpoint)
                0.01

- Simpson's Rule: use $ax^2 + bx + c$ to fit the data to calculate are

Ex: (not codes)
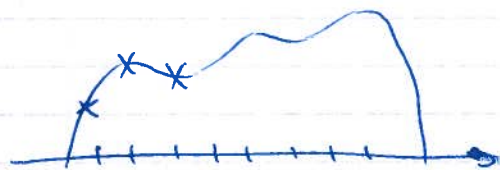
① pick 3 consecutive points   ② fit $ax^2 + bx + c$ ③ use a, b, c to find
                                                                 the area for this
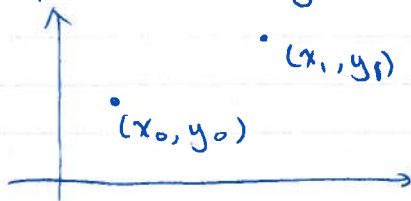   part
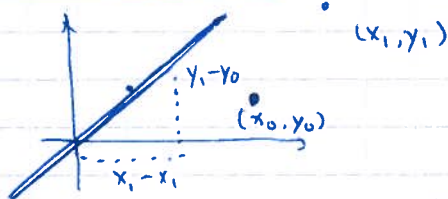
Lecture 10

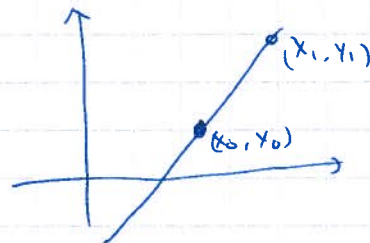- Interpolation: a technique to fit datas.
- 2 points is easy:

To fit ① we need to match the slope.

② put the line through (x_0, y_0).

Lecture 10 (Cont)

① Step 1: find slope $\dfrac{Y_0 - Y_1}{X_0 - X_1}$

② $y = \left(\dfrac{Y_0 - Y_1}{X_0 - X_1}\right) x + b \quad \leftarrow$ unknown

step 2: put the line on point (x_0, y_0)

$$y_0 = \left(\dfrac{Y_0 - Y_1}{X_0 - X_1}\right) x_0 + b$$

$$\Rightarrow b = y_0 - \dfrac{Y_0 - Y_1}{X_0 - X_1} x_0$$

so, to fit 2 point, we use funct^n

$$Y = \left(\dfrac{Y_0 - Y_1}{X_0 - X_1}\right) x + \left[Y_0 - \dfrac{Y_0 - Y_1}{X_0 - X_1} x_0\right] \qquad \text{same as } \underline{pg\ 7} \text{ in lecture 10}$$

— more than 2 points:

$$f(x) = \sum_i \omega_i y_i \quad ; \quad \omega_i = \prod_{j \neq i} \dfrac{x - x_j}{x_i - x_j}.$$

This fitting is 'stupid'. observe if $x = x_i$, the $\omega_i = 1$, and

$\omega_j \neq \omega_i = 0$ so, this always garentee to predict correct y_i.

EX: given data $\quad x = \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad \Big\}$ gives x_1, outcome is y_1.

$\qquad\qquad\qquad\quad Y = \quad y_1 \quad y_2 \quad y_3 \quad y_4$

$$f(x) = \omega_1 y_1 + \omega_2 y_2 + \omega_3 y_3 + \omega_4 y_4 \; ; \quad \omega_1 = \left(\dfrac{x - x_2}{x_1 - x_2}\right)\left(\dfrac{x - x_3}{x_1 - x_3}\right)\left(\dfrac{x - x_4}{x_1 - x_4}\right)$$

$\vdots$

Then $\quad f(x_3) = 0 \cdot y_1 + 0 \cdot y_2 + 1 \cdot y_3 + 0 \cdot y_4 = y_3.$

Error Rate:

$$\int_a^b f(x)\, dx = \sum_{k=1}^{N} \underbrace{\int_{x_{k-1}}^{x_k} f(x)\, dx}_{\text{divide } \int \text{ into N pieces.}}$$

$$= \tfrac{1}{2} h \underbrace{\sum_{k=1}^{N} \left[f(x_{k-1}) - f(x_k)\right]}_{\text{This is Simpson prediction.}} + \tfrac{1}{4} h^2 \left[f'(a) - f'(b)\right] + \dfrac{1}{12 h^3} \sum_{k=1}^{N} \left[f''(x_{k-1}) + f''(x_k)\right] + O(h_4)$$

This is Trapezoid.

So, error is about $\sim \tfrac{1}{4} h^2$

we say $O(h^2)$

This is error from Simpson. $O(h^4)$