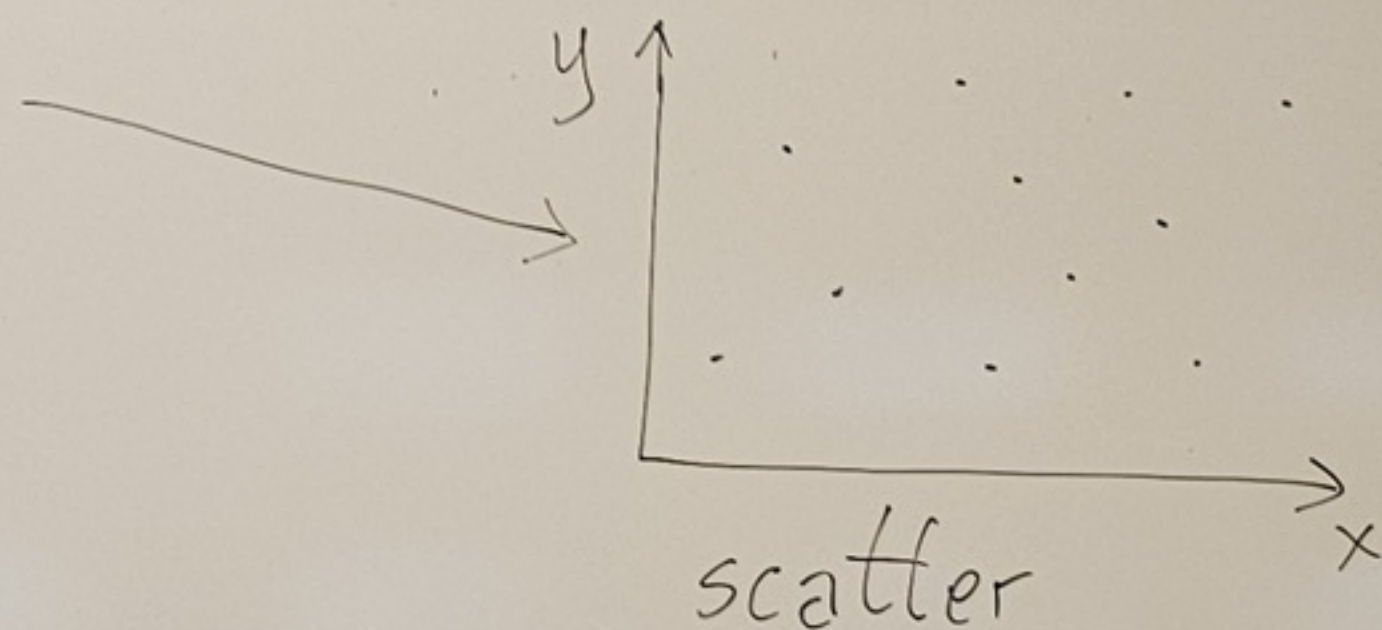
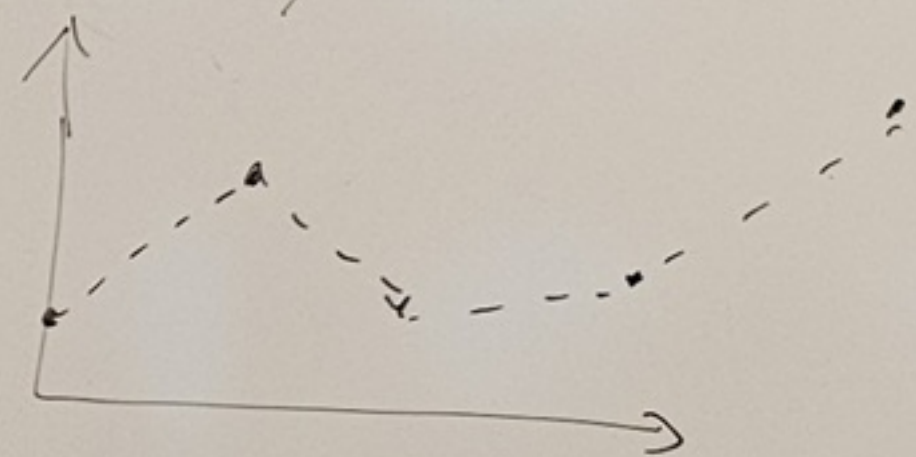
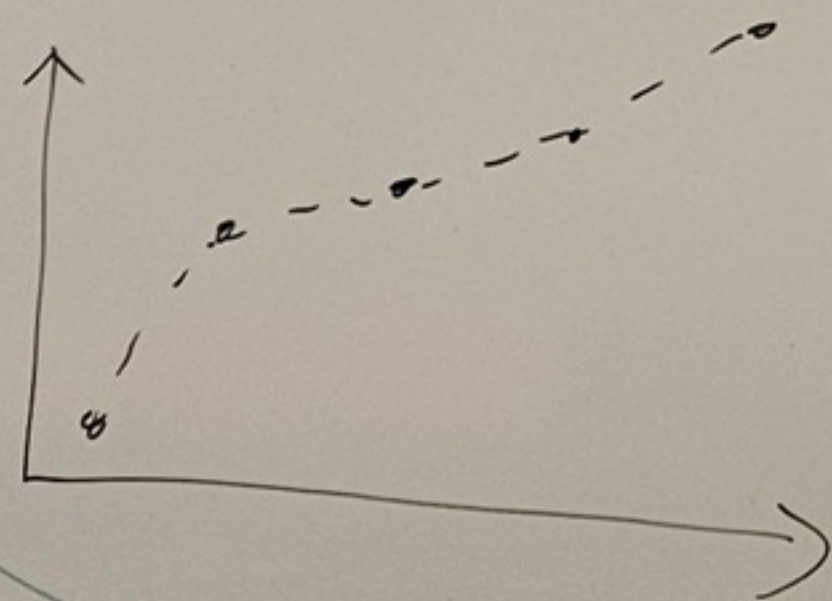


import random



{ plot(..., ..., '.', linestyle='dashed') }

skumulowane



[4, 1, 3, 4]

[4, 5, 8, 15]

4

1

3

4

Kod źródłowy ma być elastyczny

1 1 2 3 5
1 1/2 2/3 3/5

[sprawdzian za 2 tyg. +
z wydmuszek / starterów]

starter
wprint
N pierwszych liczb Fibonacciego
na listach z numpy.array

with open (mode='w')

x.txt

1	3.14
1	1.28
2	3.41
3	0.02

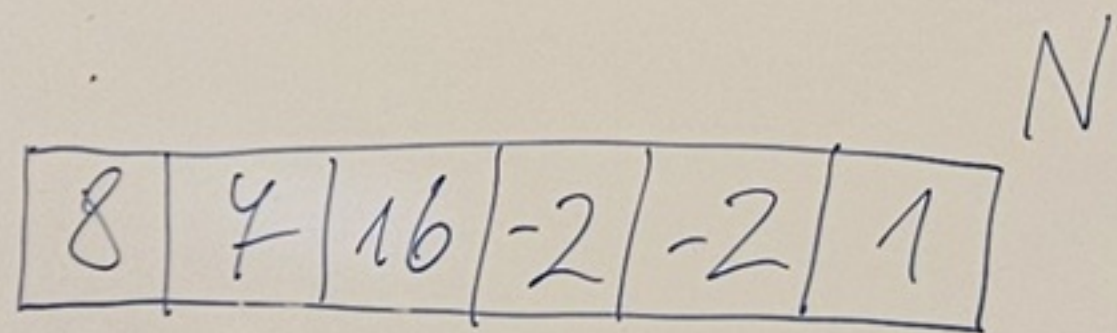
rand

$$\frac{f_i}{f_{i+1}}$$

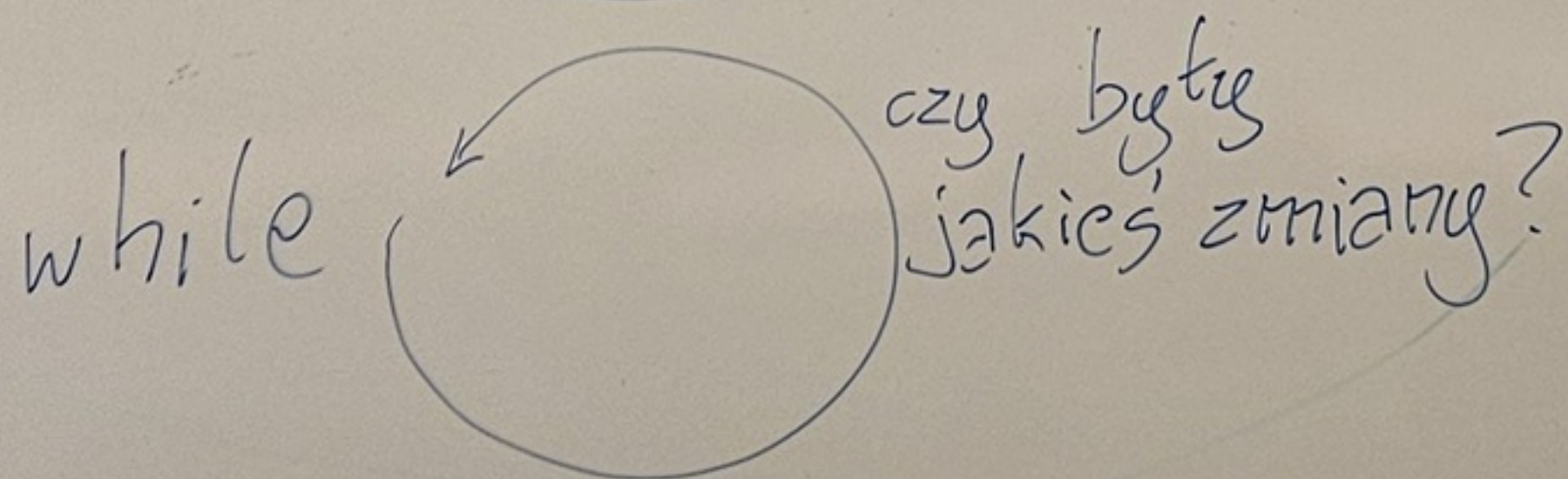
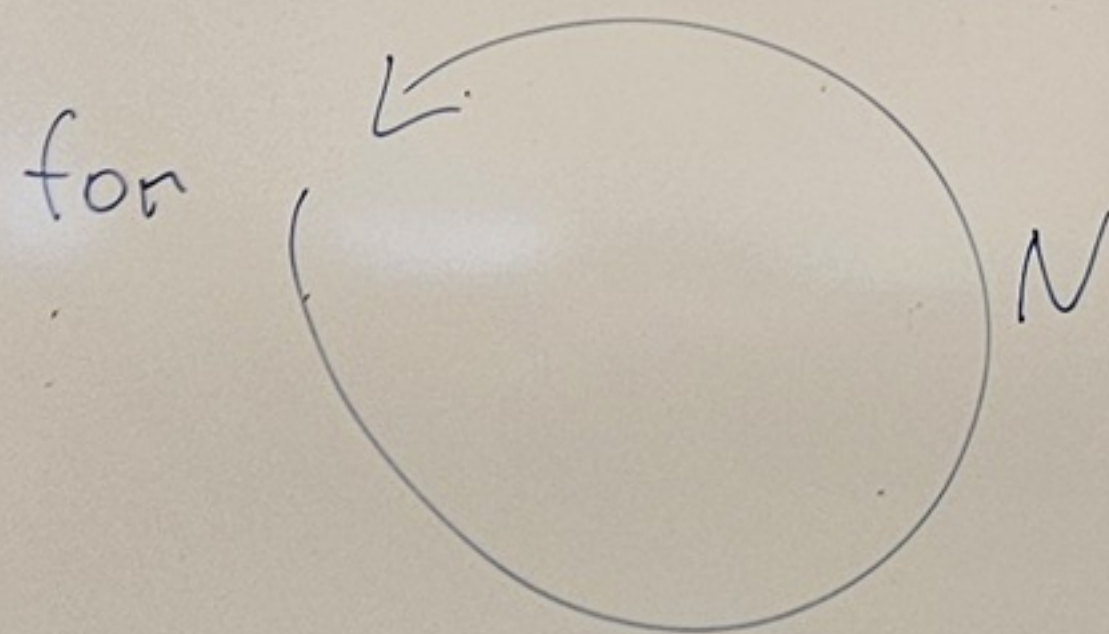
with open (mode='r')
do listy krotek

"jestem_tu".split('_')[(1, 3.14), (1, 1.28), (2, 3.41), ...]

Sort. bańbelkowe (zak. rosnąco)



for
zmień sąsiadów jeżeli trzeba



chyba mi działa

jak testować kod

detektor = True

w pętli ← detektor = False
wykryto zdarzenie → detektor = True

while → warunek kontynuacji
→ warunek zatrzymania pętli

Wyszukiwanie liniowe min
indeks_najmniejszego = 0
for każdy element
czy jest najmniejszy?

zwróć indeks najmniejszego

Siatka M wierszy, N kolumn
samiych zer

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

numpy

jako lista list
append

[0 for x in range(4)]

def macierz_zer(n_wierszy,
n_kolumn):

return

list(range(7, -1, -1))

w domu

sort przez wstaw.

-2	8	7	16	-2	1
----	---	---	----	----	---

na listach

na tabelach

w miejscu lub na nowych tabelach

a ← 2

b ← 3

c ← a

a ← b

b ← c

a, b = b, a

Wyszukiwanie liniowe min
 indeks_najmniejszego = 0
 for każdy element
 czy jest najmniejszy?

zwróć indeks najmniejszego

Siatka M wierszy, N kolumn
 samych zer

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

numpy

jako lista list
 append

[0 for x in range(4)]

def macierz_zer(n_wierszy,
 n_kolumn):

return

1			
	1		0
		1	
0		1	
			1

for ...
 for ...
 if ...

1	0	1	0	1
0	1	0	1	0

1	1	1
0	1	1
0	0	1

1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5

1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4