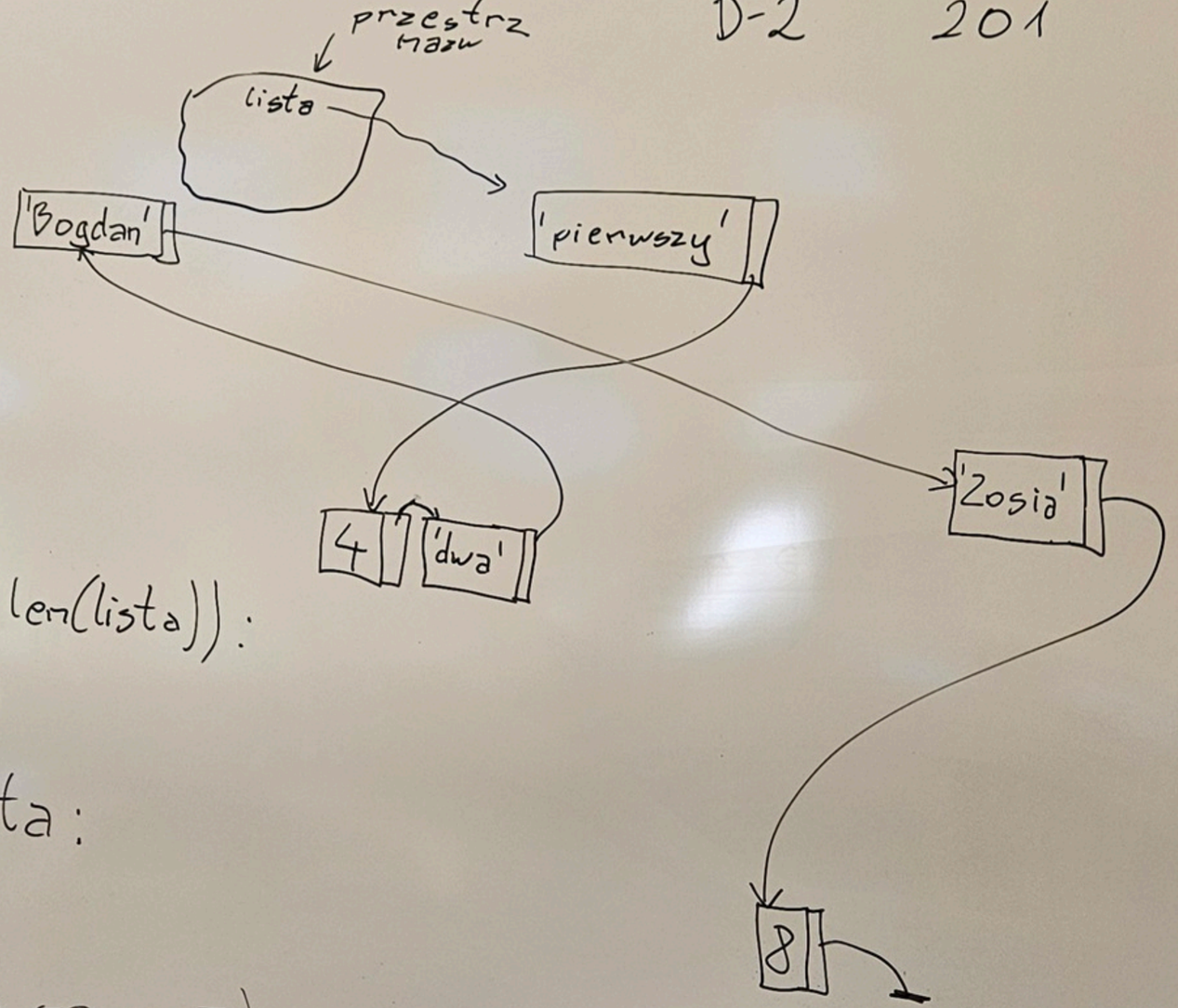
LISTY a tabele

D-2 201

```
lista = list()
lista = []
lista.append('pierwszy')
lista.append(4)
```



RAM

```
print(
 lista[3])
```

```
for i in range(len(lista)):
    print(lista[i])
```

```
print(lista[4])
```

```
for elem in lista:
    print(elem)
```

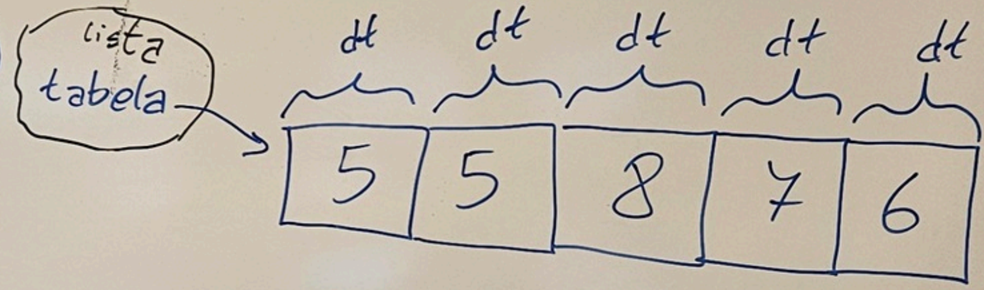
```
tętno.apppend(62)
                (58
                (74)
```

```
lista.extend([62,58,74])
```

```
import numpy as np

tab = np.array([5, 5, 8, 7, 6], dtype=float)
```

np.float64
'float'

przestrz nazw

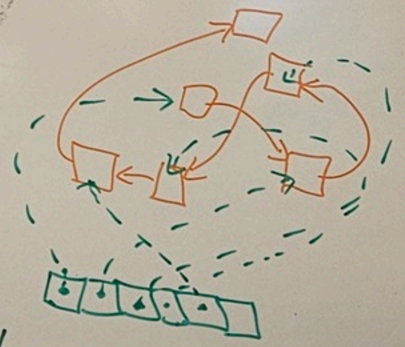


z góry wiem ~~5.0, 5, 8~~
ile elem
typ elem.

indeksowanie

```
tab[3]
tab[1]
```

id(lista)



Li[4]

```
np.array(['a', 'bb', 'ccc'])
```

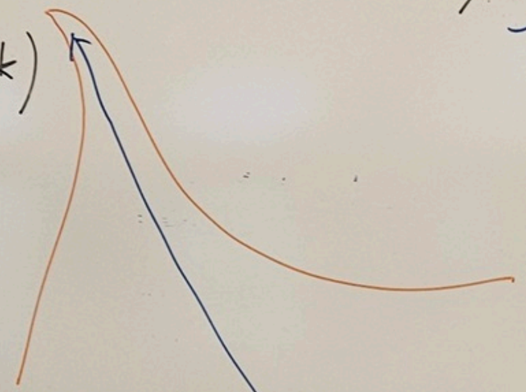
L

```
T = np.array(L)
T
```

eksperymentujemy!

czy T i L dają mi różny dostęp/interfejs do tego samego obszaru pamięci?

```
import matplotlib.pyplot as plt
import matplotlib.image as im

obrazek = im.imread('obrazek.jpg')

plt.imshow(obrazek)
plt.axis('off')
plt.show()
```



png

```
obrazek = im.imread('obrazek.png')
obrazek = 255 * obrazek
obrazek = obrazek.astype('int')
```

SPRAWDZIĆ

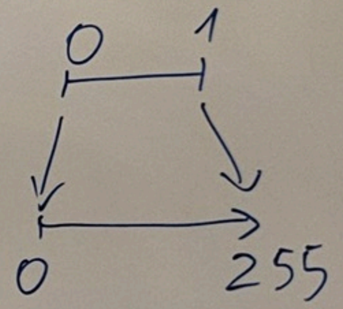
```
obrazek.shape
```

(290, 174, 3)

wys szer kanały

```
R = obrazek[:, :, 0]
G =            1
B =            2
```

[0.1 0.28 ...]



```
fig, ax = plt.subplots(1, 3)
```
figsize = (5, 10)

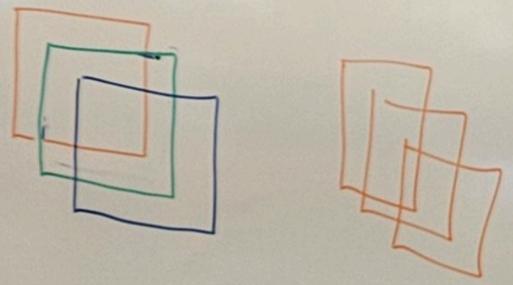
teoria

praktyka

```
ax[0].imshow(np.stack([R, R, R], axis=2)
ax[0].axis('off')
ax[1].imshow(.....[G, G, G]
ax[1]
ax[2]
ax[2]
plt.show()
```

[300, 200, 3)

obrazek



np.array([R, R, R])

plt.imshow( )

np.stack([R, R, R], axis=2).shape

eksperyment

Anaconda Pytkom

W domu sign Up

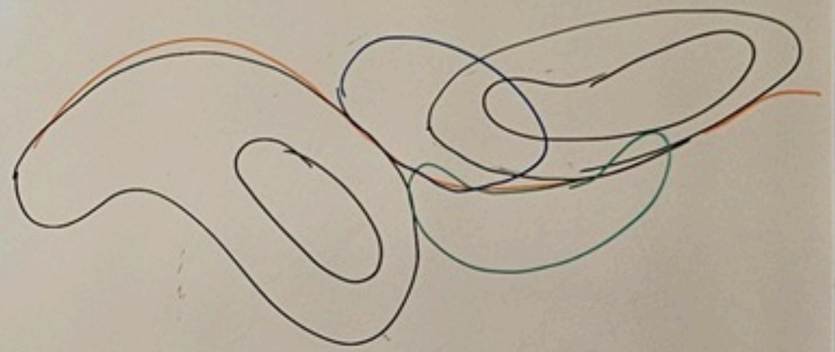
255 - R

plt.imshow (coś)
plt.show()
imshow (coś2)

plt.show()

obiekt

R G B
R B G
B G R
B R G
G B R
G R B



np.stack

np.zeros_like(R)

W domu
zrobić własną wersję

Anaconda Pytkom

W domu sign Up

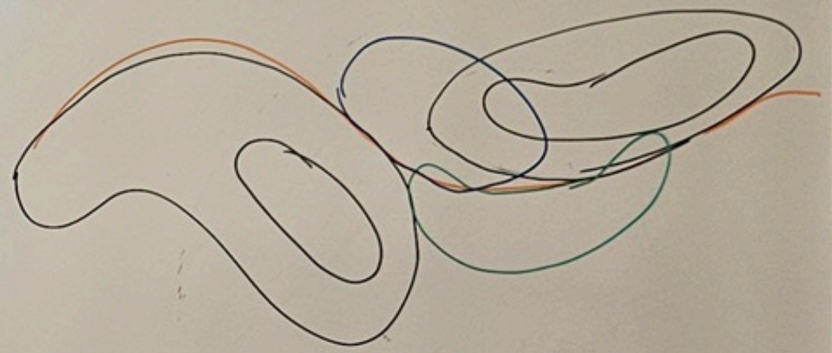
255 - R

plt.imshow (coś)
plt.show()
imshow (coś2)

plt.show()

obiekt

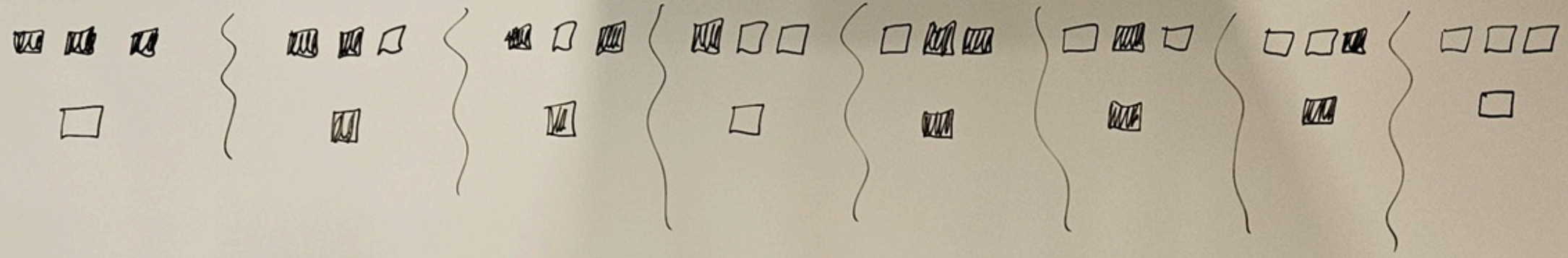
R G B
R B G
B G R
B R G
G B R
G R B

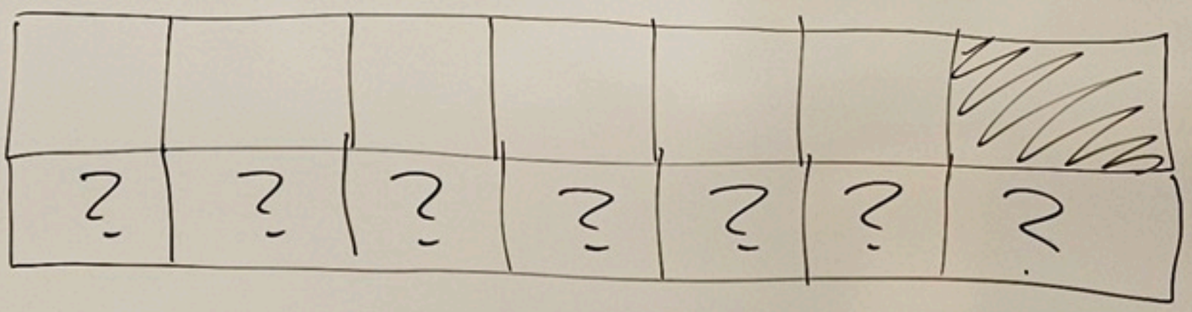


np.stack

np.zeros_like(R)

W domu
zrobić własną wersję





lista.append(coś)

lista.append(inna lista)

listazer = 6*[0]

nowa_lista = [0] + lista + [0]

```
for i in range(4):
    pass  #uzupełnię potem
if coś:
    pass
def f(x):
    pass
```

```
for i in range(7):
    rób coś
```

⇕

```
i = 0
while i < 7:
    rób coś
    i += 1
```

PROSTE EKSPERYMENTY ! ODPOWIEDZIAŁ NA WĄTPLIWOŚCI

Inicjatywa

```
import matplotlib.pyplot as plt

plt.imshow([[1, 0, 1]], cmap='binary')
plt.show()
```

```
if coś1:
    rób 1
elif coś2:
    rób2:
elif coś3:
    rób3
else:
    rób X
```

```
for.....
    for....
        if
            for...
        else:
            if...
                for
```

```
if coś and coś or coś:
    rób to
```

Listy składane (ang. list comprehension)

```
lista = []
for i in range(8):
    lista.append(i+1)
print(lista)
```

<=>

```
lista = [(i+1) for i in range(8)]
```

```
ile_elementow = 5
```

1 4 9 16 25 36 49

```
lista = [(i+1)**2 for i in range(ile_elem)
         if i % 2 == 0]
```

(2*(i+1))**2
(i+1)*(i+1)

range(1, ile_elementów, 2)

```
def gen_liste(ile_elementow):
    return .....

nowa_lista = gen_liste(ile_elementow=100)
```

PEP8

~~wolf~~

```
import time
import numpy as np

num = 10 ** 6
tab = np.arange(num)
lista = list(range(num))

start = time.time()

wynik1 = tab ** 2
stop = time.time()
czas_wykonania1 = stop - start

wynik2 = [x**2 for x in lista]
print(f'Czas wyk dla tab = {czas_wykonania1:0.5f}')
```

np.arange(6)

list(range(6))

np.array(range(6))

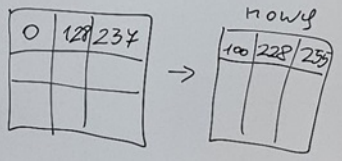
por. z tworz. listy z append()

#śle(alok.pam)

time.time()

<0, 255>

wartość = 100



- nie znamy słowa „pech"
- jak coś jest skomplikowane, to łatwo się plącze i trudno to rozplątać

wartość -100

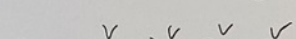
- programuję drobnymi krokami, bez przeskoków,
- najlepiej: piszę kawałeczek i od razu sprawdzam

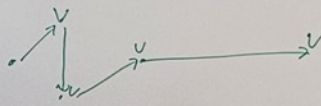
type(coś)

int
uint
uint16
uint8

def podbij(kanał, wartość): (np. R, G, lub B)
    nowy_kanał . . . .
    . . . .

zwróć nowy_kanał

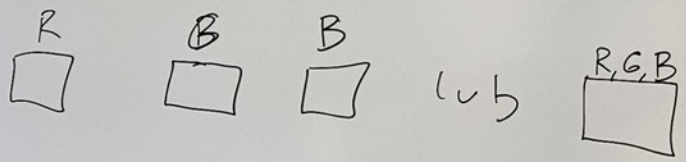
- 



bytes.pl → ALO Pwr → snake.jpg

R, G, B ←

R_{nowy} = podbij(R, 100)



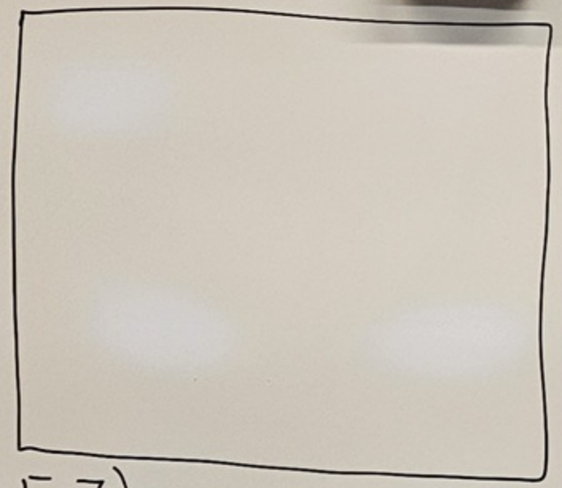
- staram się uzyskać wynik takimi środkami, które są dostępne

```
tab_nowy = tab.astype('int')
```

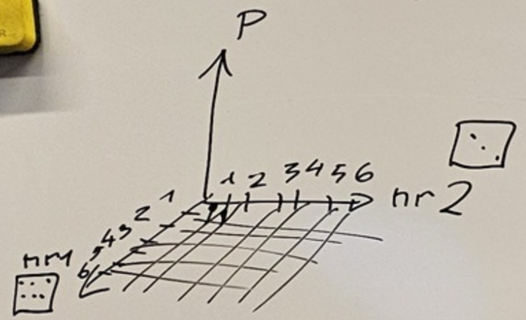
```
print(..G [42][121])
print(podbij(G..,200) [42][121])
```

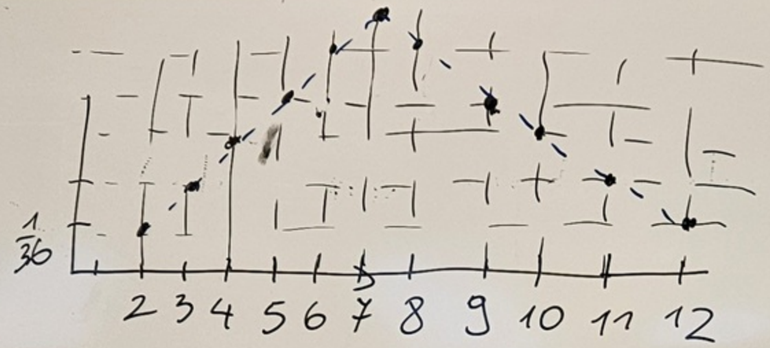
testowanie programu

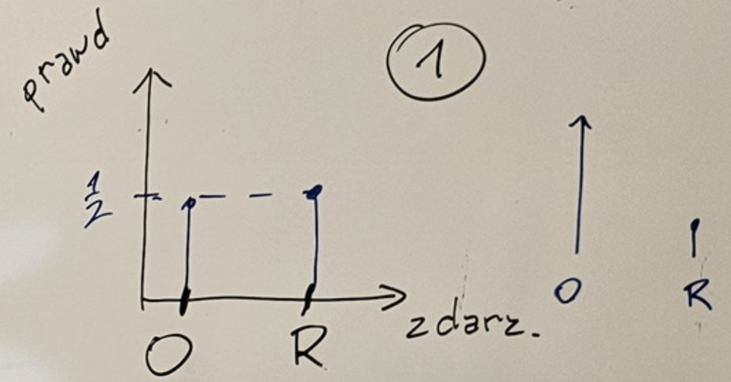
ma być 255

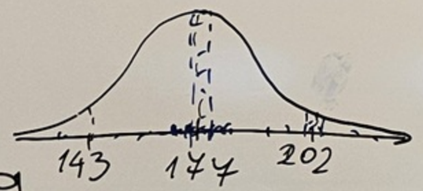


```
import numpy as np

tab = np.random.choice(['orzet', 'reszka'], p=[0.1, 0.9],
                       size=10)

tab[tab == 'orzet'][int(elem) for elem in lista]

type(tab[0])

np.random.randint(1, 7, size=10).astype('float')

['orzet', 'reszka', 'kant'], p=[0.49, 0.49, 0.02]

sum(tab == 'kant')

[i for i in range(1,7) if i != 3]

[i for i in range(1,50) if i not in [3, 13, 28, 42]]
```





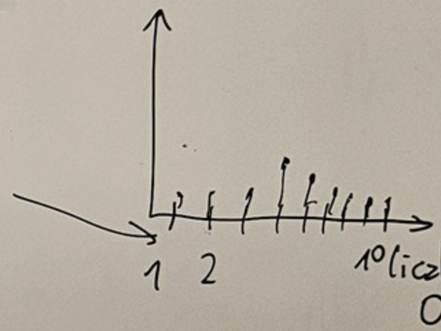




$0,9999\ldots$

$177,000400$ lim

ORRR...
RORR...
RRORR...



N rzutów

O - liczba orłów

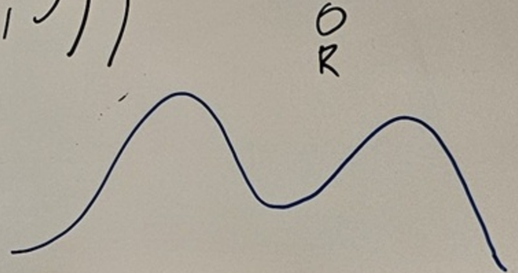
R = N - O

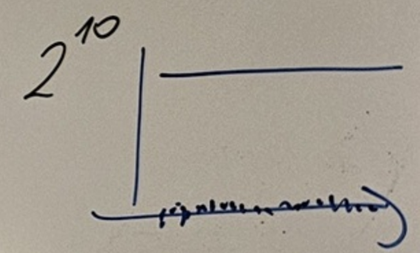
```
np.random.randint(1, 7, size=(9, 9))
```

```
obr=np.random.randint(0, 256, size=(800, 600, 3))
                                    80  60
```

```
plt.imshow(obr, interpolation='bicubic')
plt.show()
```

plt.axis('off')

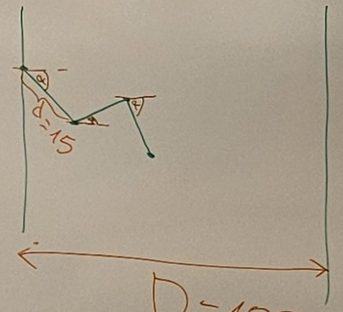




```
from numpy import random as rnd
import numpy as np

rnd.uniform(-1, 1, size=10)

rnd.uniform(0, 2*np.pi, size=40)
```

```
licznik=1
los = rnd.uniform(-1,1)
while los>0:
    print(los, licznik)
    los = rnd.uniform(-1,1)
    licznik += 1
```

neutron shielding




```
raport = [(52, 7, 'nie', 1),
          (112, 6, 'tak', 2)]

while (los := rnd.uniform(-1,1)) > 0:
    print(los)

licznik = licznik + 1
'linia \n druga'
f'N = {N}'
```

and or

N skoków = 10

przebyta odległ.	ile skoków	czy_przeszedł	nr neutronu
	8	'tak'	1
			2
			3
			4
			5

```
from numpy import random as r

r.uniform(0,1)
```

wynik

akumulator	ile prób
1.4	3
średnia	średnia

to jest pojedynczy eksperyment

1. akumulator ← 0
2. Dodaj los ∈ ⟨0,1⟩
3. Jeżeli akumulator > 1 to STOP
4. Powtarzaj eksperyment

np.