język programowania — abstrakcja „umawiamy się na zestaw symboli i wyrażeń" → opis co ma robić maszyna

niedeklaratywny ≈ imperatywny

j. niskiego poziomu
„zrozumiałe" dla maszyny — (czytelne)
ASEMBLER
?
?

j. wys. poziomu
„zrozumiałe" dla człowieka
- C++
- Java
- Python

j. deklaratywne
programista opisuje wynik, a silnik sam decyduje jak to wykonać
- SQL

rekurencja

funkcyjne
program to zestaw wywołujących się funkcji, nie ma zmiennych
- Haskell

logiczne
programista opisuje fakty lub reguły, a maszyna „wyciąga wnioski"
- Prolog

kod źródłowy — kompilator — program — działający program

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

kompilator A → kod pośredni → kompilator B → 1101100111010

kompilator ultymatywny

parser – analiza składni

np. JVM...

wirtualny ≈ umowny

W DOMU
uzupełnij nazwę jęz. i zobacz przykładowe kody

```
pow = { 'Wrocław': 320,
        'Warszawa':
        'Radom':
        'Paryż':
        'Stolec':
        'Meksyk':
        'Sworne Gacie':
        'Poznań'
      }
```

```
lud = { 'Wrocław': 900 000
        'Warszawa': - - - -
                    - - -
                    .
                    .
                    .
                    .
                    .
                    .
      }
```

```
miejsowość = 'Wrocław'
print(f" " "
w miejsowości {miejscowość}.
miesza {lud[miejscowość]} ludu
na powierzchni {pow[miejscowość]} km kwadratowych.
" " "
```

```
for klucz, wartos in pow.items():
    print(f'Miasto {klucz} ma ....')
```

W domu
wypracować wspólną „tablicę" na kody.

w domu
zobacz, jak wyglądają pliki JSON

```
pobor = ['Wroctaw', 'Cidaris', 'Pipidówa']
ile = 1000
```

w domu
dokończ, jeżeli nie udało się na zajęciach

```
pow
lud

pow['Cidaris'] = 26
lud['Cidaris'] = 3 811

pow.items()

for krotka in pow.items():
    print(krotka)
    print(f'. Miasto {krotka[0]} ma powierz....')
```

```
list(pow.keys())

for elem in pow.keys():
    print(elem)

pow.values()

for wartosc in pow.values():
    print(wartosc)
```

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

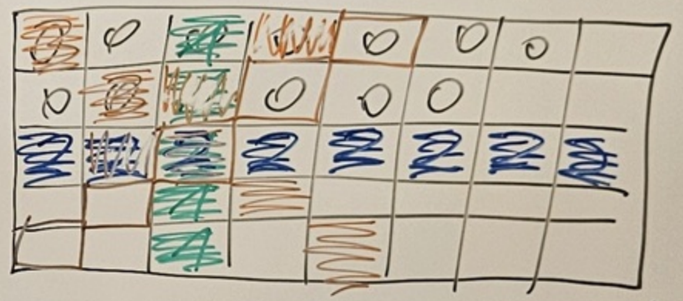
$$trA = \sum_{i=1}^{M} a_{ii}$$

$$\frac{trA}{2} = ? \frac{a_{11} + a_{22}}{2}$$

min (nrows, ncols)
max

$A_{M \times M}$ $M=2$ tab = np.zeros((5, 8))

plt.imshow(tab, cmap='nazwa') nazw_r



nrows, ncols = tab.shape

col = 2

for i in range(tab.shape[0]):
    tab[i] = 1

row = 2

mapa kolorów

0	1	2	3	4	5	6	. . .	100

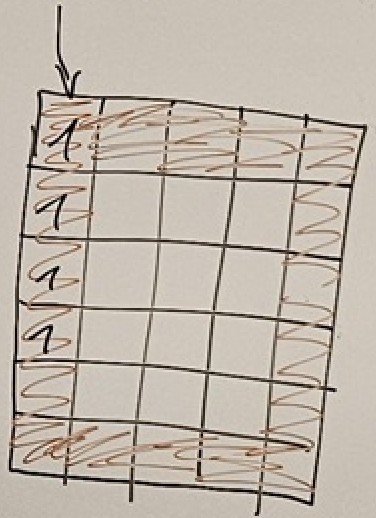
żółty [255, 0, 255]

jasnożółty [128, 0, 255]

for i in range(nrows-1, -1, -1):

x=3, y=4

kod_koloru



gen_ramke(lwier, lkol,
kod_koloru=1)

lwierszy = 3
lkolumn = 4

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

return tab

obrazek= gen_ramke(lwier=4, lkol=8)
plt.imshow(obrazek)

```
1       7 8 9
2   ·   6 1 2
3   8   5 4 3
4 5 6 7
```

https:// qlo-kody.ggorg.xyz

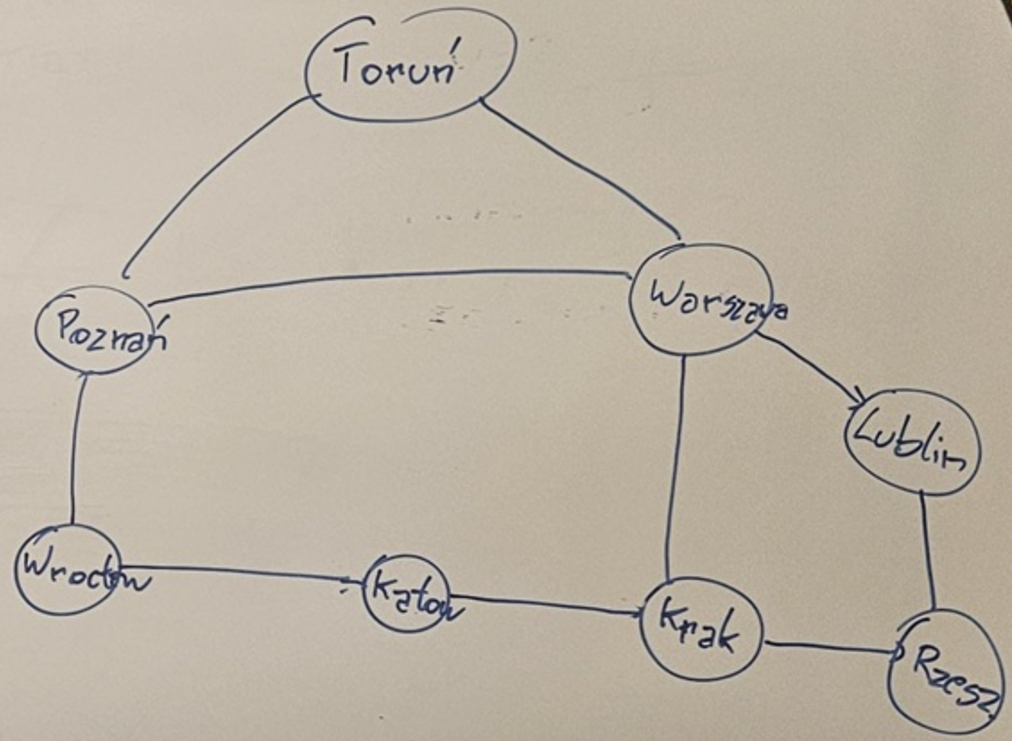
rejestracja konta: https:// go.ggorg.xyz / qlo-kody-rejestracja

1) venv

2) pip install pyvis

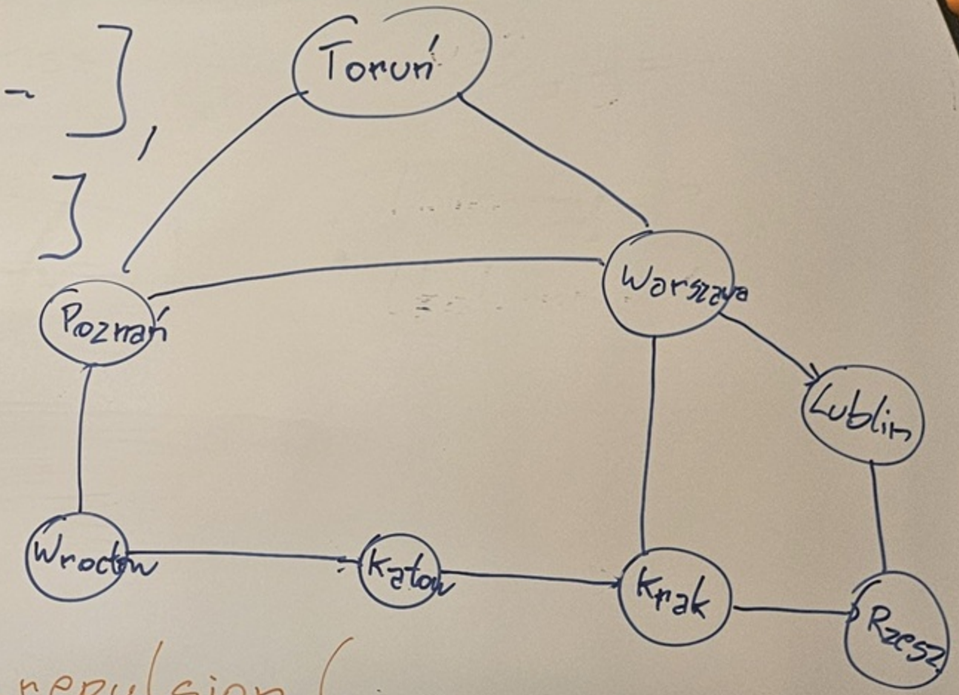
.venv → Lib

```
from pyvis.network import Network
net = Network(directed=False)
net.add_node('Wroctaw')
net.add_node('Rzeszów')
```




```
net.save_graph('moja_siec.html')
```

```
net.add_nodes([ 'Poznań', 'Toruń', 'Warszawa',... ],
              color=[ 'red' , 'lightgreen', 'black',.... ]
net.add_edges(
    [
        ('Wroctaw', 'Poznań'),
        ('Wroctaw', 'Katowice'),
        ⋮
    ])
```




```
from pyvis.network import Net
net = Network(directed=False)
net.add_node('Wroctaw')
net.add_node('Rzeszów')
```

```
net.repulsion(
    node_distance=100,
    central_gravity=0.2,
    spring_length=200
    spring_strength=0.05,
    damping=0.09
)
```

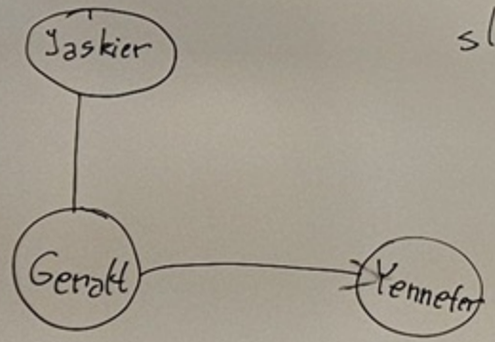
```
net.add_edges(
    [
        ('Wroctaw', 'Poznań'),
        ('Wroctaw', 'Katowice'),
        ⋮
    ])
```

```
from pyvis.network import Netw
net = Network(directed=False)
net.add_node('Wroctaw')
net.add_node('Rzeszów')
```

```
net.show_buttons(filter_=['nodes', 'edges'])
net.show_buttons(filter_='physics')
```

lub

```
# net.repulsion(
#     node_distance=100,
#     central_gravity=0.2,
#     spring_length=200
#     spring_strength=0.05,
#     damping=0.09
# )
```




```
slo={ 'Geralt':['Jaskier',
                'Yennefer'
                'Yarpen']
      'Jaskier':['Geralt',
                 'Loetlenhove']
        ⋮
      }

lista=[('Geralt','Yarpen'),('Geralt',
'Jaskier')]
```

- Jak opuszczam zajęcia, to nadrabiam w domu, żeby „być na bieżąco"
- Ćwiczyć w domu korzystanie z izolowanego środowiska (venv)

#