Faculty of Geodesy
and Cartography
WARSAW UNIVERSITY OF TECHNOLOGY

Informatyka Geodezyjna - wykłady/ćwiczenia, rok akad. 2021-2022

SciPy – ekosystem bibliotek do obliczeń i analiz inzynierskich – Numpy

Kinga Węzka
kinga.wezka@pw.edu.pl
Zakład Geodezji i Astronomii Geodezyjnej

Warsaw University
of Technology

Plan

1. Ekosystem SciPy (SciPy Stack)
2. NumPy – struktury tablicowe i obliczenia numeryczne
   - ndarray – tablice NumPy
   - ndarray – tablice NumPy – tworzenie tablic
   - array vs ndarray
   - ndarray – tablice NumPy – inspekcja
   - ndarray – tablice NumPy – referencja vs kopia
   - matrix – macierze NumPy – w przyszłości brak wsparcia!
   - NumPy – podstawowe mechanizmy działania – pomoce
3. SciPy – zaawansowane metody obliczeń numerycznych
4. SciPy v. NumPy
5. Pandas – struktury danych i narzędzia do analizy
6. Literatura

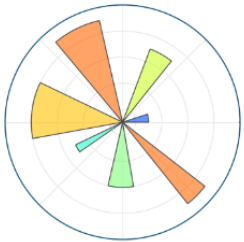
Ekosystem SciPy (SciPy Stack)

- **NumFocus** to fundacja, której celem jest wspiranie oprogramowania typu *open source* do celów obliczeń i analiz matematycznych do zastosowań inzynierskich: https://numfocus.org/community/mission
- **SciPy** (wymawiane „Sigh Pie") to oparty na Pythonie ekosystem oprogramowania typu *open source* dla matematyki, nauk technicznych. W szczególności są to niektóre z podstawowych pakietów: numpy, scipy, matplotlib, pandas etc.
  Więcej: https://numfocus.org/sponsored-projects

- **NumPy** (ang. *Numerical Python*) – zaawansowane metody obliczeń numerycznych, algebra liniowa, więcej: `www.numpy.org/`
- **SciPy** (ang. *Scientific Python*) – zaawansowane obliczenia numeryczne oparte na tablicach NumPy, wiecej: `www.scipy.org/`
- **Pandas** – prezentacja graficzna wyników (wykresy) więcej: `www.pandas.pydata.org/`

NumPy

SciPy



Matplotlib

- **Pandas** – struktury danych i narzędzia do analizy. Rozszerzeniem tej biblioteki jest **GeoPandas**, wspomagająca analizy geoprzestrzenne, więcej: `www.pandas.pydata.org/`
- **Xarray** – łatwe w użyciu struktury danych wysokiego poziomu (np. netCDF), narzędzia analityczne do pracy z wielowymiarowymi, etykietowanymi zbiorami danych i tablicami, przetwarzanie równoległe z użyciem Dask, więcej: `www.xarray.pydata.org`
- **Dask** – umożliwia bibliotekom (NumPy, pandas i scikit-learn) skalowalne obliczeń na dużych zbiorach danych – pozwala na przetwarzanie równoległe, umożliwia skalowanie od procesorów wielordzeniowych do klastrów z tysiącami węzłów, wiecej: `www.dask.org`

pandas

xarray

Dask

- **AstroPy** – popularne obliczenia stronomiczne, mechanika orbit etc., więcej: `www.astropy.org`
- **GDAL** (*Geospatial Data Abstraction Library*) – dostęp i tłumaczenie rastrowych i wektorowych formatów danych geoprzestrzennych – teledetekcja, więcej: `www.gdal.org`
- **scikit-learn** – uczenie maszynowe, wiecej: `www.scikit-learn.org`
- **scikit-image** – cyfrowe przetwarzanie obrazów, wiecej: `www.scikit-image.org`

Astropy

GDAL

scikit-learn

scikit-image

NumPy – struktury tablicowe i obliczenia numeryczne

NumPy

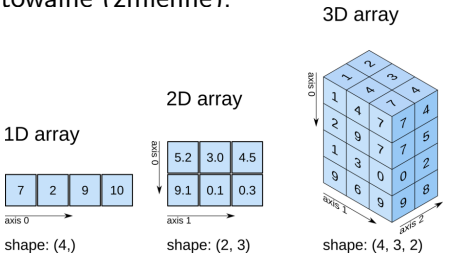
- **NumPy** opiera się na bibliotekach BLAS i LAPACK (fortran) co zwieksza wydajności obliczeń algebry liniowej.
- Podstawową funkcjonalności NumPy jest jego typ zmiennych tablicowych (`ndarray`), wykorzystywany dla n-wymiarowej tablicy jako struktury danych.
- Typ zmiennych tablicowych NumPy (`ndarray`) rozszerza język Pythona o wydajną strukturę danych przydatną do prac matematycznych i numerycznych.
- Oficjalna strona: `http://www.numpy.org/`
- Manual: `https://numpy.org/devdocs/reference/index.html` `https://numpy.org/devdocs/user/quickstart.html`

Użycie biblioteki wymaga jej instalacji i importu:

In[1]:

```
1 import numpy as np
```

- Tablice Numpy – `ndarray` – to kolekcje, które **przechowują dane tego samego typu**. Nie są tak elastyczne jak listy (różne typy), ale ze względu na jednolity i znany typ przechowywanych danych, można łatwo obliczyć jaki będzie rozmiar tablicy w pamięci.
- Numpy wykonuje operacje na całych tablicach (np. macierze), a nie na pojedyńczych elementach (ja w list). `ndarray` posiadają zoptymalizowane funkcje operujące na wszystkich przechowywanych w obiekcie danych, do przetwarzania dużych zbiorów.
- Obiekty `ndarray` zachowują swój rozmiar; przy zmianie rozmiaru takiego obiektu powstaje nowy obiekt, a obiekt sprzed zmiany zostaje usunięty.
- Obiekty `ndarray` są mutowalne (zmienne).



In[2]:

```
1 a = np.array([1,2,3])
2 print('size a = ', np.size(a))
3 print('shape a = ', np.shape(a))
```

Out:

```
1 [1 2 3]
2 size a = 3
3 shape a = (3,)
```

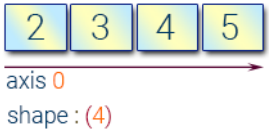
In[3]:

```
1 c = np.array([[1,2,3],[4,5,6]])
2 print('size b = ', np.size(b))
3 print('shape b = ', np.shape(b))
```

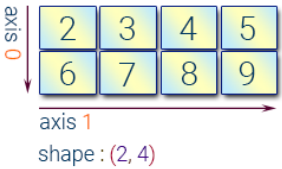
Out:

```
1 [[1 2 3]
2 [4 5 6]]
3 size b = 6
4 shape b = (2, 3)
```

Tablica 1D



Tablica 2D



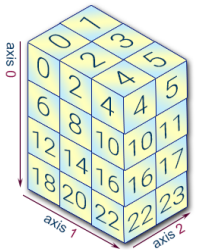
In[4]:

```
d=np.array([[(1, 11), (2, 12)], [(3,13), (4,14)]])
print('size d = ', np.size(d))
print('shape d = ', np.shape(d))
```

Out:

```
[[[ 1. 11.]
  [ 2. 12.]]
 [[ 3. 13.]
  [ 4. 14.]]]
size d = 8      shape d = (2, 2, 2)
```

Tablica 3D




```
a = np.zeros((3,4))      # tablica zer (3 wiersze, 4 kolumny)
b = np.ones((2,3))       # tablica jedynek (2 wiersze, 3 kolumny)
c = np.linspace(0, 5, 10) # utworzenie tablicy: start, stop, ilość wart.
c = np.arange(0, 5, 1)  # utworzenie tablicy: start, stop, interwał
d = np.full((2,2),7)     # tablica o wymiarach (2x2) wypełnionej 7
```

Pełna dokumentacja: numpy.org/devdocs/reference/routines.array-creation.html

`numpy.array` jest funkcją która zwraca obiekt `numpy.ndarray`. Nie ma typu obiektu nazywanego `numpy.array`

- Listę można zrzutować na typ `numpy.ndarray` korzystając z funkcji `numpy.array`

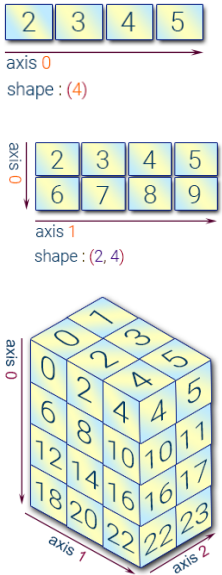
In[5]:

```
import numpy as np
a = [1,2,3]
print('type a: ', type(a))
b = np.array(a)
print('type b: ', type(b))
print(isinstance(b, (np.ndarray)))
```

Out[5]:

```
type a: <class 'list'>
        type b: <class 'numpy.ndarray'>
        True
```

- `ndarray.ndim` – liczba osi (wymiarów) tablicy.
- `ndarray.shape` – wymiary tablicy. Krotka liczb całkowitych wskazujących rozmiar tablicy w każdym wymiarze.
- `ndarray.size` – całkowita liczba elementów tablicy. Jest to równe iloczynowi elementów kształtu.
- `ndarray.dtype` – obiekt opisujący typ elementów w tablicy. Tablice mają standardowe typy Pythona lub specjalne typy NumPy np: Numpy.int32, numpy.float64.
- `ndarray.itemsize` – rozmiar w bajtach każdego elementu tablicy. Np. tablica elementów typu float64 ma 8 elementów: 8 (= 64/8), natomiast jeden z typów complex32 ma 4 elementy: 4 (= 32/8). Jest to równoważne z `ndarray.dtype.itemsize`.



- utworzenie tablicy:

In[6]:

```
a = np.array([1,2,3])
b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
d = np.ones((3,4)) # macierz jednostkowa
```

- sprawdzanie tablicy:

In[7]:

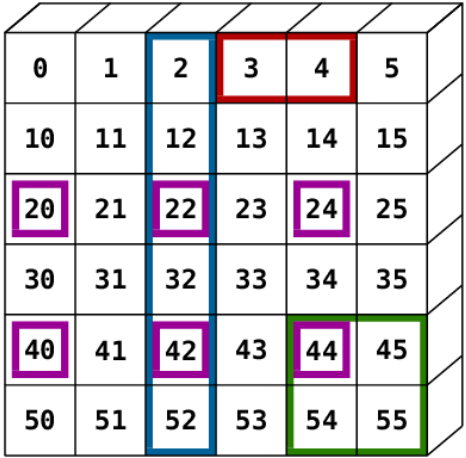
```
b.shape    # wymiar tablicy
b.ndim     # liczba wymiarów
b.dtype    # typy danych w tablicy
```

- operacje/metody tablicowe, pełna dokumentacja:
  numpy.org/devdocs/reference/routines.array-manipulation.html

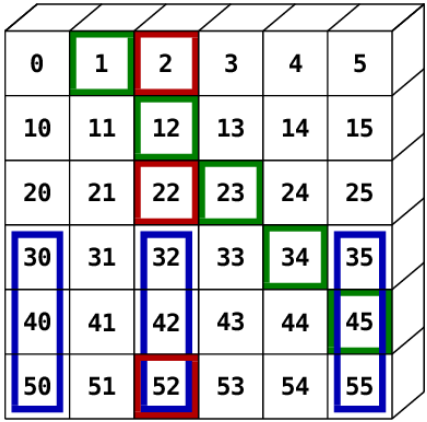
In[8]:

```
i = np.transpose(b) # macierz transponowana
np.hsplit(a,3)      # podział poziomo na miejscu 3go idx
np.insert(a, 1, 5)  # wstawianie wartości na miejsce idx
```

```
>>> a[0, 3:5]
array([3, 4])

>>> a[4:, 4:]
array([[44, 55],
       [54, 55]])

>>> a[:, 2]
a([2, 12, 22, 32, 42, 52])

>>> a[2::2, ::2]
array([[20, 22, 24],
       [40, 42, 44]])
```



```
>>> a[(0,1,2,3,4), (1,2,3,4,5)]
array([1, 12, 23, 34, 45])

>>> a[3:, [0,2,5]]
array([[30, 32, 35],
       [40, 42, 45],
       [50, 52, 55]])

>>> mask = np.array([1,0,1,0,0,1], dtype=bool)
>>> a[mask, 2]
array([2, 22, 52])
```



Należy pamietać że znak równości w Pythonie tworzy referencję, a nie kopię. Aby wykonać kopię tablicy `ndarray` należy skorzystać ze specjalnych metod.

- `B = A` tworzy referencję.
- `B = A.copy()` lub `B = A[:]` tworzy kopię.

```
1 A = np.array([[ 0, 1, 2, 3,  4],
2               [ 5,  6,  7,  8,  9],
3               [10, 11, 12, 13, 14]])
4 B = A[:2,2:]    # << REFERENCJA
5 B[0,0] = 999
6 print(A)
```

```
1 [[  0   1 999   3   4]
2  [  5   6   7   8   9]
3  [ 10  11  12  13  14]]
```

```
1 A = np.array([[ 0, 1, 2, 3,  4],
2               [ 5,  6,  7,  8,  9],
3               [10, 11, 12, 13, 14]])
4 B = A[:2,2:].copy()    # << KOPIA
5 B[0,0] = 999
6 print(A)
```

```
1 [[  0   1   2   3   4]
2  [  5   6   7   8   9]
3  [ 10  11  12  13  14]]
```

- **matrix** – dodatkowy typy zmiennych numpy dla definiowania macierzy.

In[9]:

```
A = np.array([(1.5,2,3), (4,5,6)])
B = np.matrix(A) # rzutowanie ndarray do matrix
A = np.matrix([(1.5,2,3), (4,5,6)]) # utworzenie typu matrix
```

Podstawowe różnice (NumPy_4_matlab_users.pdf)

- dla `ndarray`, zapis `A * B` oznacza iloczyn odpowiadających elementów (element-wise multiplication), natomiast `dot(A, B)` - oznacza mnożenie macierzy;
- dla `matrix`, `A * B` oznacza mnożenie macierzy;
- obiekty `ndarray` mogą mieć liczbę wymiarów > 2, `matrix` zawsze mają dwa wymiary.

`ndarray` – podstawowy typ zmiennych w NumPy, wiele funkcji przyjmuje jako argumenty typ `ndarray`, albo zwraca wynik typu `ndarray`, nawet jeżeli działania zostały wykonane na `matrix`.
`matrix` – **nie zaleca się już używania tej klasy**, nawet w przypadku algebry liniowej. Zamiast tego należy używać zwykłych tablic. Klasa zostanie usunięta w przyszłości.

Pomoc

- Podstawowe mechanizmy działania – Jupyter Notebook:
  OneDrive – tutoriale_pomoce/lib_numpy/help_numpy.ipynb
- Podstawowe równice pomiedzy Matlab a Numpy:
  https://numpy.org/doc/stable/user/numpy-for-matlab-users.html
  OneDrive – tutoriale_pomoce/lib_numpy/numpy4matlab_users.pdf
- Zestawienie najczęściej uzywanych metod:
  OneDrive – tutoriale_pomoce/lib_numpy/numpy_cheat_sheet.pdf

OneDrive – https://wutwaw-my.sharepoint.com/:f:/g/personal/kinga_wezka_pw_edu_pl/EsQqBLWPPMdKqIxCaQSn8xQBYrdo87VD8OFOzBD_v_GBXw?e=AqU8dw

SciPy – zaawansowane metody obliczeń numerycznych

SciPy

- SciPy opiera się na obiektach tablicowych NumPy ((`ndarray`)) i zawiera rozbudowany zestaw bibliotek naukowych.
- SciPy zawiera moduły do optymalizacji, algebry liniowej, integracji, interpolacji, funkcji specjalnych, FFT, przetwarzania sygnałów i obrazu, rozwiązań ODE, metody całkowania numerycznego, rozwiązywanie równań różniczkowych, i innych zadań wspólnych dla nauki i inżynierii.
- Oficjalna strona: `https://www.scipy.org/`
- Manual: `https://scipy-lectures.org`

Użycie biblioteki wymaga jej instalacji i importu:

In[10]:

```
1  import scipy as sp
```

- Funkcje specjalne (`scipy.special`)
- Całkowanie numeryczne (`scipy.integrate`)
- Optymalizacja (`scipy.optimize`)
- Interpolacja (`scipy.interpolate`)
- Transformacje Fouriera (`scipy.fftpack`)
- Przetwarzanie sygnału (`scipy.signal`)
- Algebra liniowa (`scipy.linalg`)
- Problemy z wartością własną z ARPACK
- Dane przestrzenne (`scipy.spatial`)
- Statystyka (`scipy.stats`)
- Przetwarzanie obrazu (`scipy.ndimage`)
- Plik IO (`scipy.io`)

```
1
2  # import
3  import scipy.linalg
4  scipy.linalg.lstsq(M, y)
5
6  # import
7  from scipy import linalg
8  linalg.lstsq(M, y)
9
10 # import
11 from scipy.linalg import lstsq
12 lstsq(M, y)
```

Więcej: https://docs.scipy.org/doc/scipy/reference/py-modindex.html

SciPy v. NumPy

- **NumPy** zawiera typ danych tablicowych i najbardziej podstawowe operacje: indeksowanie, sortowanie, przekształcanie, podstawowe funkcje elementarne i tak dalej.
- Cały kod numeryczny znajdowałby się w SciPy. Jednak jednym z ważnych celów NumPy jest kompatybilność, dlatego NumPy stara się zachować wszystkie funkcje obsługiwane przez jednego z jego poprzedników.
- W ten sposób NumPy zawiera pewne funkcje algebry liniowej, nawet jeśli bardziej poprawnie należą do SciPy. W każdym razie SciPy zawiera bardziej w pełni funkcjonalne wersje modułów algebry liniowej, a także wiele innych algorytmów numerycznych.
- `scipy.linalg` vs `numpy.linalg`
  - `scipy.linalg` zawiera wszystkie funkcje w `numpy.linalg` plus kilka innych bardziej zaawansowanych, których nie ma w `numpy.linalg`.
  - zaletą używania `scipy.linalg` zamiast `numpy.linalg` jest to, że zawsze jest kompilowany z obsługą BLAS / LAPACK, podczas gdy dla `numpy` jest to opcjonalne. W związku z tym `scipy` wersja może być szybsza w zależności od sposobu zainstalowania `numpy`. Dlatego, chyba że nie chcesz dodawać scipy jako zależności do swojego programu numpy, użyj `scipy.linalg` zamiast `numpy.linalg`.

Pandas – struktury danych i narzędzia do analizy

Pandas

- Wysoko-wydajne, łatwe w użyciu struktury danych i narzędzia do analizy danych dla języka programowania Python.
- Pozwala na wydajne łączenie, sortowanie i kombinacje za pomocą indeksowania kolumn
- Podstawowymi strukturami danych są:
  - `Series` jednowymiarowa tablica z etykietami, pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html
  - `DataFrame` czyli dwuwymiarowa struktura z etykietami (nazwy kolumn i wierszy), pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html
- Dokumetacja: http://pandas.pydata.org/

Użycie biblioteki wymaga jej instalacji i importu:

In[11]:

```
1 import pandas as pd
```

In[12]:

```
name = ["Marek", "Anna", "Ewa"]
age = [28, 10, 33]
married = [True, True, False]

df_from_list = pd.DataFrame(
    data=[name, age, married],
    columns=["name", "age", "married"])
```

Out[12]:

```
name  age  married
0  Marek   28     True
1   Anna   10     True
2    Ewa   33    False
```

Literatura

- NumPy www.numpy.org/
- SciPy: www.scipy.org/
- Matplotlib: www.matplotlib.org/
- Pandas: www.pandas.pydata.org/
- Xarray: www.xarray.pydata.org
- Dask: www.dask.org
- AstroPy: www.astropy.org
- GDAL: www.gdal.org
- scikit-learn: www.scikit-learn.org
- scikit-image: www.scikit-image.org

Dziękuje za uwagę

Kinga Węzka kinga.wezka@pw.edu.pl