

Wstęp do programowania w środowisku PYTHON



<https://docs.python.org/3/library/>

Kontakt: mkuich@fuw.edu.pl

Materiały: www.fuw.edu.pl/~mkuich/tik2021/

Cechy Python'a

- UWAGA! Korzystamy z Python3 (preferowany python3.6)
- Język programowania wysokiego poziomu ogólnego przeznaczenia
- Polecenia/kody w Python'ie są **interpretowane** przez specjalny interpreter
- Posiada bardzo rozbudowany pakiet bibliotek standardowych
- Dominuje paradygmat obiektowości → wszystko jest obiektem
- Automatycznie zarządza pamięcią i posiada dynamiczny system typów zmiennych
- Najpopularniejsze IDE:
 - ▶ PyCharm, www.jetbrains.com/pycharm/
 - ▶ Jupyter, www.jupyter.org
 - ▶ Visual Studio Code code.visualstudio.com/
 - ▶ Spyder, www.spyder-ide.org
 - ▶ IDLE - zintegrowany
- Na pracowni:
 - ▶ IDLE
 - ▶ Atom edytor tekstu + python3 w terminalu
 - ▶ Spyder
- Sugestia dla prywatnych PC: Anaconda + Spyder www.anaconda.com/

Uruchamianie

- “Programy” → “Programowanie” → “IDLE 3”
lub otwieramy zwykły terminal i wpisujemy `python3`
 - ▶ utworzyliśmy interpreter, w którym mamy dostęp do pełnej funkcjonalności Python’a i interaktywnie możemy pracować. Wypróbuj np.:
`print('Hello worl')` - polecenie wypisze na ekran ciąg znaków
 - ▶ taki tryb może posłużyć, jako podręczny kalkulator; oblicz:
`3+4`
`17/3`
`17%3` – modulo, reszta z dzielenia
`2**3` – funkcja wykładnicza
`1+3*4` vs `(1+3)*4` – kolejność wykonywania działań
 - ▶ zmienne w trybie interaktywnym, wypróbuj:
`a=8`
`b=10-a`
`a*b`
 - ▶ funkcje matematyczne:
`sin(3.14)` ???
brakuje pakietu
 - ▶ `quit()` lub `CTRL+D` pozwala na wyjście z programu

Python - podstawowe pakiety

❶ Pakiet math

- ▶ biblioteka z funkcjami matematycznymi
- ▶ użycie: `import math` lub `import math as m`

❷ Pakiet numpy

- ▶ biblioteka do zaawansowanych obliczeń numerycznych, operacji na macierzach
- ▶ wprowadza obiekty `array`
- ▶ użycie: `import numpy` lub `import numpy as n`

❸ Pakiet matplotlib

- ▶ biblioteka do wizualizacji danych
- ▶ tutaj: rysowanie wykresów
- ▶ użycie: `import matplotlib.pyplot` lub `import matplotlib.pyplot as pl`

❹ Pakiet scipy

- ▶ biblioteka do zastosowań naukowych
- ▶ tutaj: dopasowywanie funkcji do danych, liniowa algebra, całkowanie ...
- ▶ użycie: `import scipy.optimize` lub `import scipy.optimize as sp`

Funkcje matematyczne - pakiet math

- Podstawowe funkcje:

funkcja	opis
<code>log(wartość)</code>	logarytm, o podstawie e
<code>log10(wartość)</code>	logarytm, o podstawie 10
<code>sin(wartość)</code>	sinus, wartość w radianach
<code>cos(wartość)</code>	cosinus, wartość w radianach
<code>sqrt(wartość)</code>	pierwiastek kwadratowy
<code>pow(wartość, wartość)</code>	funkcja wykładnicza

- Dokumentacja: <https://docs.python.org/3/library/math.html>

- Przykład:

```
import math
math.sin(3.14)
```

- **Zadanie 1. Oblicz:**

$\sin(\pi/3)$

$\log_{10} 10$

$\sqrt[3]{8}$

Skrypty

- Skrypt to plik tekstowy o rozszerzeniu `.py`, zawierający kod źródłowy programu interpretowanego przez interpreter
- Uruchamianie skryptu przez interpreter:
IDLE: `Run`→`Run Module` (F5)
terminal: `python3 skrypt.py`
- Opcjonalnie jeśli dodamy shebang w pierwszej linijce naszego skryptu, to możemy go uruchomić w terminalu następująco:
`./skrypt.py`
→ shebang: `#!/usr/bin/python3`
→ należy pamiętać, że skrypt powinien mieć uprawnienia do wykonywania
- Skrypty edytujemy w dowolnym edytorze tekstu lub IDE
- Przykładowy skrypt:

```
#!/usr/bin/python3  
print('Hello world')
```

To wartość a nie zmienna posiada typ

- Porównaj:

```
print('2+3')
```

ciąg znaków

```
print(2+3)
```

liczby całkowite

- Przykłady typów wartości

Typ	Opis	Przykład
str	ciąg znaków (niezmienny)	'abc' lub "abc"
int	liczba całkowita o dowolnej wartości	13
float	liczba zmiennoprzecinkowa	3.1416
complex	liczba zespolona	3 + 2.7i
bool	wartość logiczna	True lub False
list	lista dowolnych wartości	[4.0, 'abc', True]
tuple	krotka (niezmienna - lista wartości)	(4.0, 'abc', True)
set	nieindeksowany zbiór wartości	{4.0, 'abc', True}
dict	słownik, czyli tablica asocjacyjna	{'key0':1.0, 'key1':False}
NoneType	obiekt reprezentujący brak wartości, odpowiednik wartości null	None

- Sprawdzanie typu wartości: `type(<wartość>)`
np.: `type(44)`

Uporządkowane sekwencje obiektów

Krotki

- ▷ **niezmienne**
- ▷ definicja przez `()`
- ▷ przykład:
`krotka1 = (1, 2, 5)`
- ▷ wypisanie całek listy:
`print(krotka1)`
- ▷ wypisanie elementu listy:
`print(krotka1[1])`
iteracja od 0 do n-1
- ▷ wypisanie długości listy:
`print(len(krotka1))`

Listy

- ▷ **zmienne**
- ▷ definicja przez `[]`
- ▷ przykład:
`lista1 = [1, 2, 5]`
- ▷ wypisanie całek listy:
`print(lista1)`
- ▷ wypisanie elementu listy:
`print(lista1[1])`
iteracja od 0 do n-1
- ▷ wypisanie długości listy:
`print(len(lista1))`
- ▷ podmiana elementu:
`lista1[0]=7`
- ▷ dodanie elementu na końcu:
`lista1.append(13)`
- ▷ usuwanie elementu:
`del lista1[2]`

Instrukcja warunkowa **if** Przykład:

```
if <warunek>:  
    <rób coś>  
elif <inny warunek>:  
    <rób coś innego>  
else:  
    <rób coś innego>
```

```
if x==2:  
    print('x wynosi dwa')  
elif x==3:  
    print('x wynosi trzy')  
else:  
    print('x jest inne')
```

Pętla **for**

```
for <element> in <sekwencja>:  
    <rób coś przy każdej iteracji pętli>
```

Przykład 1: liczby od 0 do 10

```
for i in range(0,11):  
    print(i)
```

Przykład 2: iteracja po elementach

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

Zadanie 2.

Napisz program `suma.py`, wykorzystujący pętlę `for` do liczenia sumy liczb całkowitych od 1 do 100.

Zadanie 3.

Napisz program `modulo.py`, w którym utworzysz listę dowolnych liczb całkowitych i wypiszesz na ekran te, które są podzielne przez 3. Należy skorzystać z pętli `for` i instrukcji warunkowej `if`.

Funkcje

- Funkcje definiujemy na początku programu
- Przykład definicji funkcji i jej wywołania w skrypcie:

```
def przywitanie(): #definicja funkcji
    print('Pozdrowienia z mojej funkcji!') #instrukcje

przywitanie()
```

- Przykład funkcji zwracającej wartość i obliczającej obwód trójkąta:

```
def perimeter(a, b, c):
    return a + b + c

a = float(input('Enter side a: '))
b = float(input('Enter side b: '))
c = float(input('Enter side c: '))
print('perimeter: ', perimeter(a, b, c))
```

Zadanie 4.

Napisz skrypt `funkcje.py` zawierający trzy funkcje liczące: pole koła, trójkąta i kwadratu. Wczytaj z klawiatury niezbędne dane i policz dla nich wszystkie trzy pola. Wynik wypisz na ekran.

Wskazówka: Pamiętaj o zaimportowaniu pakietu pozwalającego wykorzystywać funkcje i stałe matematyczne. Wykorzystaj instrukcje warunkowe, aby sprawdzić poprawność wymiarów figur podanych przez użytkownika, a w przypadku podania wymiarów mniejszych lub równych 0, wyświetl komunikat.

Rysowanie z kolekcji

```
import matplotlib.pyplot as p

#zbior punktow x
x0=[1,2,3,4,5]
#zbiór punktow y odpowiadajacy x
y0=[2,4,6,8,10]

p.plot(x0,y0)
p.plot(x0,y0,'ro')
#zapisywanie obrazka
p.savefig('przyklad1.pdf')
#wyświetlanie obrazka
p.show()
```

Podstawowe opcje rysowania

Other options for the color characters are:

```
'r' = red  
'g' = green  
'b' = blue  
'c' = cyan  
'm' = magenta  
'y' = yellow  
'k' = black  
'w' = white
```

Options for line styles are

```
'-' = solid  
 '--' = dashed  
 ':' = dotted  
 '-.' = dot-dashed  
 '.' = points  
 'o' = filled circles  
 '^' = filled triangles
```

Więcej na:

https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html

Zadanie 5.

Napisz skrypt `sinus1.py` zawierający definicję dwóch list bądź krotek. Pierwsza z kolekcji, x , powinna zawierać 7 punktów z przedziału od 0 do 2π , a druga, y wartości $\sin x$. Narysuj wykres sinusa zieloną ciągłą linią i zapisz do w pliku `sinus1.pdf`.

Wskazówka: Pamiętaj o zaimportowaniu pakietu pozwalającego wykorzystywać funkcje i stałe matematyczne.

Tablice z pakietu numpy

- ▷ **Przykład 1 - tworzenie i wypisanie tablicy 1D**
- ▷ Przykład 2 - tablice jako argumenty operatorów i funkcji
- ▷ Przykład 3 - tworzenie zakresów
- ▷ Przykład 4 - tworzenie i drukowanie tablic 2D
- ▷ Przykład 5 - indeksowanie oraz wycinanie wierszy i kolumn
- ▷ Przykład 6 - wczytywanie tablic dwuwymiarowych z pliku
- ▷ Przykład 7 - mnożenie wektorów

```
#dołączenie pakietu numpy
import numpy as n

#tworzenie tablicy
v = n.array([2, 5, 1, 3])
#wypisanie całej tablicy
print(v)
#wypisanie elementu tablicy
print(v[2])
```

Tablice z pakietu numpy

- ▷ Przykład 1 - tworzenie i wypisanie tablicy 1D
- ▷ **Przykład 2 - tablice jako argumenty operatorów i funkcji**
- ▷ Przykład 3 - tworzenie zakresów
- ▷ Przykład 4 - tworzenie i drukowanie tablic 2D
- ▷ Przykład 5 - indeksowanie oraz wycinanie wierszy i kolumn
- ▷ Przykład 6 - wczytywanie tablic dwuwymiarowych z pliku
- ▷ Przykład 7 - mnożenie wektorów

```
import numpy as n

def f(x):
    return n.sin(x * n.pi / 180)

v = n.array([2,5,1,3])
#mnożenie przez skalar
print(2 * v)
#dodawanie element do elementu
print(v + n.array([1,2,-2,-1]))
#mnożenie elementu przez element
print(v * n.array([1,2,-2,-1]))
#przekazanie tablicy do funkcji
print(f(v))
```

Tablice z pakietu numpy

- ▷ Przykład 1 - tworzenie i wypisanie tablicy 1D
- ▷ Przykład 2 - tablice jako argumenty operatorów i funkcji
- ▷ **Przykład 3 - tworzenie zakresów**
- ▷ Przykład 4 - tworzenie i drukowanie tablic 2D
- ▷ Przykład 5 - indeksowanie oraz wycinanie wierszy i kolumn
- ▷ Przykład 6 - wczytywanie tablic dwuwymiarowych z pliku
- ▷ Przykład 7 - mnożenie wektorów

```
import numpy as n
```

```
#jeśli znamy interwał między punktami  
u = n.arange(0, 10, 1.5)  
print(u)
```

```
#jeśli znamy liczbę punktów  
t = n.linspace(0.0, 10.0, 100)  
print(t)
```

Tablice z pakietu numpy

- ▷ Przykład 1 - tworzenie i wypisanie tablicy 1D
- ▷ Przykład 2 - tablice jako argumenty operatorów i funkcji
- ▷ Przykład 3 - tworzenie zakresów
- ▷ **Przykład 4 - tworzenie i drukowanie tablic 2D**
- ▷ Przykład 5 - indeksowanie oraz wycinanie wierszy i kolumn
- ▷ Przykład 6 - wczytywanie tablic dwuwymiarowych z pliku
- ▷ Przykład 7 - mnożenie wektorów

```
import numpy as n

m = n.array([[1,2,3],[4,5,6]])
print(m)

#mozna dodawać kolejne wymiary ta-
blicy w nieskończoność
```

Tablice z pakietu numpy

- ▷ Przykład 1 - tworzenie i wypisanie tablicy 1D
- ▷ Przykład 2 - tablice jako argumenty operatorów i funkcji
- ▷ Przykład 3 - tworzenie zakresów
- ▷ Przykład 4 - tworzenie i drukowanie tablic 2D
- ▷ **Przykład 5 - indeksowanie oraz wycinanie wierszy i kolumn**
- ▷ Przykład 6 - wczytywanie tablic dwuwymiarowych z pliku
- ▷ Przykład 7 - mnożenie wektorów

```
import numpy as n

m = n.array([[1,2,3],[4,5,6]])
print(m)
print(m[0, 1])
print(m[0, :])
print(m[:, 1])
# arr[wiersz, kolumna]
# iteracja od 0 do n-1
# : oznacza cały wiersz (bądź kolumnę)
```

Tablice z pakietu numpy

- ▷ Przykład 1 - tworzenie i wypisanie tablicy 1D
- ▷ Przykład 2 - tablice jako argumenty operatorów i funkcji
- ▷ Przykład 3 - tworzenie zakresów
- ▷ Przykład 4 - tworzenie i drukowanie tablic 2D
- ▷ Przykład 5 - indeksowanie oraz wycinanie wierszy i kolumn
- ▷ **Przykład 6 - wczytywanie tablic dwuwymiarowych z pliku**
- ▷ Przykład 7 - mnożenie wektorów

```
import numpy as n

tab = n.loadtxt('array.txt')
print(tab)
#wyswietlanie pierwszej kolumny
print(tab[:, 0])
#wyswietlanie drugiej kolumny
print(tab[:, 1])
#wyswietlanie trzeciej kolumny
print(tab[:, 2])
```

Tablice z pakietu numpy

- ▷ Przykład 1 - tworzenie i wypisanie tablicy 1D
- ▷ Przykład 2 - tablice jako argumenty operatorów i funkcji
- ▷ Przykład 3 - tworzenie zakresów
- ▷ Przykład 4 - tworzenie i drukowanie tablic 2D
- ▷ Przykład 5 - indeksowanie oraz wycinanie wierszy i kolumn
- ▷ Przykład 6 - wczytywanie tablic dwuwymiarowych z pliku
- ▷ **Przykład 7 - mnożenie wektorów**

```
import numpy as np

v = np.array([1, 2, 3])
u = np.array([4, 5, 6])

skalary = np.dot(v,u)
wektorowy = np.cross(v,u)

print(skalarny)
print(wektorowy)
```

Zadanie 6.

Napisz skrypt, który wczyta dane z pliku `array2.txt` do tablicy z pakietu `numpy`, sprawdź jej wymiary. Następnie w dokumentacji pakietu `numpy` znajdź funkcję, która pozwoli Ci na utworzenie macierzy jednostkowej o wymiarach odpowiadających wczytanej tablicy. Oblicz iloczyn elementów odpowiadających w obu macierzach, iloczyn skalarny obu tablic, a także iloczyn wektorowy macierzy `array2` i macierzy jednostkowej oraz odwrotny. Wszystkie wyniki wypisz na ekranie.

Rysowanie z tablicy - przykład 8.1

```
import matplotlib.pyplot as p
import numpy as n

#zbior punktow x
x1 = n.array([1,2,3,5,8])
#zbiór punktów y
y1 = n.array([6,9,7,8,7])

p.plot(x1,y1)
p.savefig('przyklad8.pdf')
p.show()
```

Zadanie 7.

Napisz skrypt `sinus2.py` zawierający definicję dwóch tablic. Pierwsza z tablica, `x`, powinna zawierać 100 punktów z przedziału od 0 do 2π , a druga, `y` wartości $\sin x$. Narysuj wykres sinusa niebieską przerywaną linią i zapisz do w pliku `sinus2.pdf`. Powstały w wyniku makra wykres porównaj z wykresem `sinus1.pdf`

Wskazówka: Pamiętaj o zaimportowaniu pakietu pozwalającego wykorzystywać funkcje i stałe matematyczne.

Rysowanie wykresów - kontynuacja przykładów

- ▷ **Przykład 8.1 - rysowanie z tablicy**
- ▷ **Przykład 8.2 - wykreślanie funkcji (zadanie 7)**
- ▷ Przykład 8.3 - rysowanie danych z pliku bez niepewności
- ▷ Przykład 8.4 - formatowanie wykresu
- ▷ Przykład 8.5 - rysowanie danych z pliku z niepewnościami
- ▷ Przykład 8.6 - legenda
- ▷ Przykład 8.7 - tekst
- ▷ Przykład 8.8 - osie
- ▷ Przykład 8.9 - siatka, opisy i tytuły

```
import matplotlib.pyplot as p
import numpy as n
import math as m

x1 = n.array([1,2,3,5,8])
y1 = n.array([6,9,7,8,7])
p.plot(x1,y1)

x2 = n.linspace(0.0,2*m.pi,100)
y2 = n.sin(x2)
p.plot(x2, y2)

p.savefig('przyklad8.pdf')
```

Rysowanie wykresów - kontynuacja przykładów

- ▷ Przykład 8.1 - rysowanie z tablicy
- ▷ Przykład 8.2 - wykreślanie funkcji (zadanie 7)
- ▷ **Przykład 8.3 - rysowanie danych z pliku bez niepewności**
- ▷ Przykład 8.4 - formatowanie wykresu
- ▷ Przykład 8.5 - rysowanie danych z pliku z niepewnościami
- ▷ Przykład 8.6 - legenda
- ▷ Przykład 8.7 - tekst
- ▷ Przykład 8.8 - osie
- ▷ Przykład 8.9 - siatka, opisy i tytuły

```
import matplotlib.pyplot as p
import numpy as n
import math as m

x1 = n.array([1,2,3,5,8])
y1 = n.array([6,9,7,8,7])
p.plot(x1,y1)

x2 = n.linspace(0.0,2*m.pi,100)
y2 = n.sin(x2)
p.plot(x2, y2)

data = n.loadtxt('plot.txt')
x3 = data[:, 0]
y3 = data[:, 1]
p.plot(x3, y3)

p.savefig('przyklad8.pdf')
```

- ▷ Przykład 8.1 - rysowanie z tablicy
- ▷ Przykład 8.2 - wykreślanie funkcji (zadanie 7)
- ▷ Przykład 8.3 - rysowanie danych z pliku bez niepewności
- ▷ **Przykład 8.4 - formatowanie wykresu**
- ▷ Przykład 8.5 - rysowanie danych z pliku z niepewnościami
- ▷ Przykład 8.6 - legenda
- ▷ Przykład 8.7 - tekst
- ▷ Przykład 8.8 - osie
- ▷ Przykład 8.9 - siatka, opisy i tytuły

```
import matplotlib.pyplot as p
import numpy as n
import math as m

x1 = n.array([1,2,3,5,8])
y1 = n.array([6,9,7,8,7])
p.plot(x1,y1,'r:',linewidth = 6)

x2 = n.linspace(0.0,2*m.pi,100)
y2 = n.sin(x2)
p.plot(x2, y2, 'b- -')

data = n.loadtxt('plot.txt')
x3 = data[:, 0]
y3 = data[:, 1]
p.plot(x3, y3,'go')

p.savefig('przyklad8.pdf')
```

- ▷ Przykład 8.1 - rysowanie z tablicy
- ▷ Przykład 8.2 - wykreślanie funkcji (zadanie 7)
- ▷ Przykład 8.3 - rysowanie danych z pliku bez niepewności
- ▷ Przykład 8.4 - formatowanie wykresu
- ▷ **Przykład 8.5 - rysowanie danych z pliku z niepewnościami**
- ▷ Przykład 8.6 - legenda
- ▷ Przykład 8.7 - tekst
- ▷ Przykład 8.8 - osie
- ▷ Przykład 8.9 - siatka, opisy i tytuły

```
import matplotlib.pyplot as p
import numpy as n
import math as m

x1 = n.array([1,2,3,5,8])
y1 = n.array([6,9,7,8,7])
p.plot(x1,y1, 'r:', linewidth = 6)

x2 = n.linspace(0.0,2*m.pi,100)
y2 = n.sin(x2)
p.plot(x2, y2, 'b - -')

data = n.loadtxt('plot.txt')
x3 = data[:, 0]
y3 = data[:, 1]

dx3 = data[:, 2]
dy3 = data[:, 3]
p.errorbar(x3,y3,xerr=dx3,yerr=
           dy3,capsize=3,fmt='o')

p.savefig('przyklad8.pdf')
```

- ▷ Przykład 8.1 - rysowanie z tablicy
- ▷ Przykład 8.2 - wykreślanie funkcji (zadanie 7)
- ▷ Przykład 8.3 - rysowanie danych z pliku bez niepewności
- ▷ Przykład 8.4 - formatowanie wykresu
- ▷ Przykład 8.5 - rysowanie danych z pliku z niepewnościami
- ▷ **Przykład 8.6 - legenda**
- ▷ Przykład 8.7 - tekst
- ▷ Przykład 8.8 - osie
- ▷ Przykład 8.9 - siatka, opisy i tytuły

```
...  
x1 = n.array([1,2,3,5,8])  
y1 = n.array([6,9,7,8,7])  
l1,=p.plot(x1,y1,'r:',linewidth=6)  
  
x2 = n.linspace(0.0,2*m.pi,100)  
y2 = n.sin(x2)  
l2,=p.plot(x2, y2, 'b - -')  
  
data = n.loadtxt('plot.txt')  
x3 = data[:, 0]  
y3 = data[:, 1]  
dx3 = data[:, 2]  
dy3 = data[:, 3]  
l3 = p.errorbar(x3,y3,xerr=dx3,  
                yerr=dy3,capsize=3, fmt='o')  
  
p.legend((l1, l2, l3), ('array',  
                        'function', 'file'))  
p.savefig('przyklad8.pdf')
```

Rysowanie wykresów - kontynuacja przykładów

- ▷ Przykład 8.1 - rysowanie z tablicy
- ▷ Przykład 8.2 - wykreślanie funkcji (zadanie 7)
- ▷ Przykład 8.3 - rysowanie danych z pliku bez niepewności
- ▷ Przykład 8.4 - formatowanie wykresu
- ▷ Przykład 8.5 - rysowanie danych z pliku z niepewnościami
- ▷ Przykład 8.6 - legenda
- ▷ **Przykład 8.7 - tekst**
- ▷ Przykład 8.8 - osie
- ▷ Przykład 8.9 - siatka, opisy i tytuły

...

```
p.legend((l1, l2, l3), ('array',  
                        'function', 'file'))
```

```
p.text(1,4, 'ąęłńóśź')  
p.text(1, 5, r'
```

```
    $\Delta=\frac{1}{  
r^2}\frac{\partial}{\partial r}  
r^2\frac{\partial}{\partial r}+  
\frac{1}{r^2}\left(\frac{1}{\sin  
\theta}\frac{\partial}{\partial  
\theta}\sin\theta\frac{\partial}{\partial  
\theta}+\frac{1}{  
\sin^2\theta}\frac{\partial^2}{\partial\phi^2}\right)$  
    ')
```

```
p.savefig('przyklad8.pdf')
```

Rysowanie wykresów - kontynuacja przykładów

- ▷ Przykład 8.1 - rysowanie z tablicy
- ▷ Przykład 8.2 - wykreślanie funkcji (zadanie 7)
- ▷ Przykład 8.3 - rysowanie danych z pliku bez niepewności
- ▷ Przykład 8.4 - formatowanie wykresu
- ▷ Przykład 8.5 - rysowanie danych z pliku z niepewnościami
- ▷ Przykład 8.6 - legenda
- ▷ Przykład 8.7 - tekst
- ▷ **Przykład 8.8 - osie**
- ▷ Przykład 8.9 - siatka, opisy i tytuły

...

```
p.axis([0.1, 10, -2, 10])
```

```
p.savefig('przyklad8.pdf')
```

→ pierwsza i druga liczba określają zakres osi X

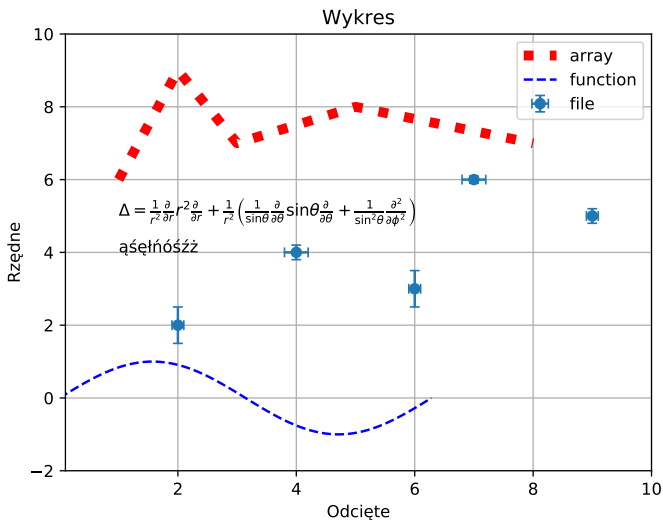
→ kolejne dwie liczby określają zakres osi Y

Rysowanie wykresów - kontynuacja przykładów

- ▷ Przykład 8.1 - rysowanie z tablicy
- ▷ Przykład 8.2 - wykreślanie funkcji (zadanie 7)
- ▷ Przykład 8.3 - rysowanie danych z pliku bez niepewności
- ▷ Przykład 8.4 - formatowanie wykresu
- ▷ Przykład 8.5 - rysowanie danych z pliku z niepewnościami
- ▷ Przykład 8.6 - legenda
- ▷ Przykład 8.7 - tekst
- ▷ Przykład 8.8 - osie
- ▷ **Przykład 8.9 - siatka, opisy i tytuły**

```
...  
  
pl.grid(True)  
  
p.xlabel('Odcięte')  
p.ylabel('Rzędne')  
  
p.title('Wykres')  
  
p.savefig('przyklad8.pdf')
```

“Przykładowy wykres”



Wzmianka o subplot'cie

```
import numpy as n
import matplotlib.pyplot as pl
def f(t):
    return n.exp(-t) * n.cos(2*n.pi*t)

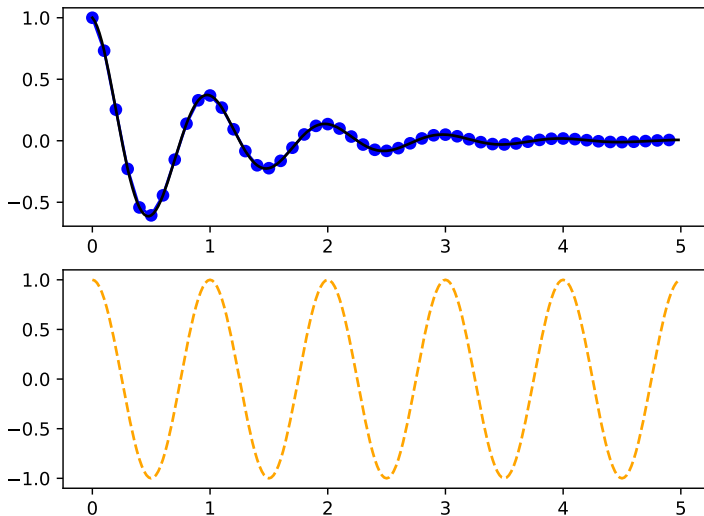
t1 = n.arange(0.0, 5.0, 0.1)
t2 = n.arange(0.0, 5.0, 0.02)

p.figure()
p.subplot(211) # nrows, ncols, index
p.plot(t1, f(t1), color='blue', marker='o')
p.plot(t2, f(t2), color='black')

p.subplot(212) # nrows, ncols, index
p.plot(t2, n.cos(2*n.pi*t2), color='orange', linestyle='- -')

p.savefig('subprzyklad8.pdf')
```

Subplot



Fitowanie funkcji do danych - pakiet SciPy

- Pakiet SciPy jest biblioteką do zastosowań naukowych
- Tutaj wykorzystanie podbiblioteki `optimize`:

```
import scipy.optimize lub import scipy.optimize as sp
```

```
https://docs.scipy.org/doc/scipy/reference/optimize.html
```

- Zastosujemy metodę `curve_fit` do dopasowania funkcji do danych

```
import scipy.optimize as sp
```

```
par,cov = sp.curve_fit(f,x,y,p0=start,sigma=s,  
absolute_sigma=True)
```

```
from scipy.optimize import curve_fit
```

```
par,cov = curve_fit(f,x,y,p0=start,sigma=s,  
absolute_sigma=True)
```

Curve_fit

- Wywołanie funkcji dopasowującej krzywą teoretyczną do danych:

```
par,cov = curve_fit(f,x,y,p0=start,sigma=s,absolute_sigma=True)
```

- Metoda zwraca:
 - ⇒ macierz 1D z parametrami (**par**) dopasowania funkcji **f**
 - ⇒ 2D macierz kowariancji (**cov**) dopasowania
- Argumenty metody **curve_fit**:
 - ⇒ **f** - dopasowywana funkcja
 - ⇒ **x** - macierz ze współrzędnymi x punktów
 - ⇒ **y** - macierz ze współrzędnymi y punktów
 - ⇒ **p0** - macierz z początkowymi wartościami parametrów*
 - ⇒ **sigma** - macierz z niepewnościami współrzędnych y*
 - ⇒ **absolute_sigma** - informuje, czy niepewności y są podane w jednostkach absolutnych lub względnych*
 - * = parametry opcjonalne

https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html

Dopasowanie funkcji do danych - przykłady

- ▷ **Przykład 9.1 -
wykreślanie danych z
pliku, bez niepewności**
- ▷ Przykład 9.2 -
dopasowywanie prostej
oraz drukowanie wartości
i błędów parametrów
- ▷ Przykład 9.3 - wykreślanie
dopasowanej zależności
- ▷ Przykład 9.4 - zadawanie
wartości początkowych
parametrów fitu
- ▷ Przykład 9.5 -
uwzględnianie błędów na
osi rzędnych

```
import numpy as n
import matplotlib.pyplot as pl

data = n.loadtxt('line.txt')
x = data[:, 0]
y = data[:, 1]

pl.plot(x, y, 'o')

pl.savefig('przyklad9.pdf')
```

Dopasowanie funkcji do danych - przykłady

- ▷ Przykład 9.1 - wykreślanie danych z pliku, bez niepewności
- ▷ **Przykład 9.2 - dopasowywanie prostej oraz drukowanie wartości i błędów parametrów**
- ▷ Przykład 9.3 - wykreślanie dopasowanej zależności
- ▷ Przykład 9.4 - zadawanie wartości początkowych parametrów fitu
- ▷ Przykład 9.5 - uwzględnianie błędów na osi rzędnych

```
...  
from scipy.optimize import curve_fit  
data = n.loadtxt('line.txt')  
x = data[:, 0]  
y = data[:, 1]  
pl.plot(x, y, 'o')  
  
def f(x, a, b):  
    return a * x + b  
  
par, cov = curve_fit(f, x, y)  
  
# wypisanie parametrów fitu  
print(par)  
  
# wypisanie niepewności parametrów fitu  
print(n.sqrt(n.diag(cov)))  
  
pl.savefig('przyklad9.pdf')
```

Dopasowanie funkcji do danych - przykłady

- ▷ Przykład 9.1 - wykreślanie danych z pliku, bez niepewności
- ▷ Przykład 9.2 - dopasowywanie prostej oraz drukowanie wartości i błędów parametrów
- ▷ **Przykład 9.3 - wykreślanie dopasowanej zależności**
- ▷ Przykład 9.4 - zadawanie wartości początkowych parametrów fitu
- ▷ Przykład 9.5 - uwzględnianie błędów na osi rzędnych

...

```
#przygotowanie współrzędnych x z danego zakresu
```

```
xpar = n.arange(-10, 10, 0.1)
```

```
#rysowanie dopasowanej zależności
```

```
pl.plot(xpar, f(xpar,par[0],par[1]))
```

```
#pl.plot(xpar, f(xpar, *par))
```

```
pl.savefig('przyklad9.pdf')
```

Dopasowanie funkcji do danych - przykłady

- ▷ Przykład 9.1 - wykreślanie danych z pliku, bez niepewności
- ▷ Przykład 9.2 - dopasowywanie prostej oraz drukowanie wartości i błędów parametrów
- ▷ Przykład 9.3 - wykreślanie dopasowanej zależności
- ▷ **Przykład 9.4 - zadawanie wartości początkowych parametrów fitu**
- ▷ Przykład 9.5 - uwzględnianie błędów na osi rzędnych

```
...  
  
par, cov = curve_fit(f, x, y,  
                    p0=n.array([0.7, 0.5]))  
  
print(par)  
print(n.sqrt(n.diag(cov)))  
  
xpar = n.arange(-10, 10, 0.1)  
pl.plot(xpar, f(xpar, par[0], par[1]))  
  
pl.savefig('przyklad9.pdf')
```

Dopasowanie funkcji do danych - przykłady

- ▷ Przykład 9.1 - wykreślanie danych z pliku, bez niepewności
- ▷ Przykład 9.2 - dopasowywanie prostej oraz drukowanie wartości i błędów parametrów
- ▷ Przykład 9.3 - wykreślanie dopasowanej zależności
- ▷ Przykład 9.4 - zadawanie wartości początkowych parametrów fitu
- ▷ **Przykład 9.5 - uwzględnianie błędów na osi rzędnych**

```
...  
data = n.loadtxt('line.txt')  
x = data[:, 0]  
y = data[:, 1]  
s = data[:, 2]  
pl.errorbar(x,y,yerr=s,fmt='o')  
  
def f(x, a, b):  
    return a * x + b  
  
par,cov=curve_fit(f, x, y,  
    p0 = n.array([0.7,0.5]),  
    sigma = s,  
    absolute_sigma = True)  
  
...  
  
pl.savefig('przyklad9.pdf')
```

Histogramy: Przykład 10.1 - wykreślanie histogramu

```
import matplotlib.pyplot as pl
import numpy as n

data = n.loadtxt('wahadlo.txt')

pl.hist(data)

pl.xlabel('T[s]')
pl.ylabel('N')
pl.title('Okres drgan wahadla')

pl.savefig('przyklad10.pdf')
```

Histogramy: Przykład 10.2 - fitowanie histogramu

```
...
data = n.loadtxt('wahadlo.txt')
#pl.hist(data)
num, bins, patches = pl.hist(data)

bincenters=[]
for i in range(0, len(bins)-1):
    bincenters.append(0.5*(bins[i]+bins[i+1]))

#bincenters=0.5*(bins[1:]+bins[:-1])

def gauss(x, mu, var, p):
    return p*n.exp(-((x-mu)**2)/(2*var*var))

par, cov = curve_fit(gauss, bincenters, num)
x = n.linspace(3.1, 3.5, 50)
pl.plot(x,gauss(x, *par), 'g-')
...
pl.savefig('przyklad10.pdf')
```

Zadania przygotowujące do kolokwium:

www.fuw.edu.pl/~mkuich/tik2021/python/zadania_python.pdf

Poniżej dodatek dla chętnych:

Liczby pseudo-losowe

- Generacja N całkowitych liczb pseudo-losowych z przedziału $[a, b)$:
`data=numpy.random.randint(a,b,N)`
- Generacja N rzeczywistych liczb losowych z przedziału $[a, b)$:
`data=a+(b-a)*numpy.random.random_sample(N)`
- Dostępne jest losowanie liczb pseudo-losowych z różnych rozkładów, np.:
 - ▶ jednorodnego (jak wyżej)
 - ▶ dwumianowego
 - ▶ trójkątnego
 - ▶ normalnego
 - ▶ poissona
 - ▶ ...

<https://docs.scipy.org/doc/numpy-1.15.0/reference/routines.random.html>

Liczby pseudo-losowe i histogramy (1)

```
import numpy as n
import matplotlib.pyplot as pl

a=5
b=25
data=n.random.randint(a,b,100000)

y,binEdges = n.histogram(data,bins=10)
bincenters=0.5*(binEdges[1:]+binEdges[:-1])
err=n.sqrt(y)
# szerokosc slupka w histogramie
w=0.9*(b-a)/10.
# Zakres poziomej osi
pl.xlim(5,26)
# przygotowanie histogramu z błędami
pl.bar(bincenters,y,color='g',width=w,yerr=err)
pl.title("Liczby losowe wygenerowane z płaskiego
rozkladu")
pl.show()
```

Liczby pseudo-losowe i histogramy (2)

```
import numpy as n
import matplotlib.pyplot as pl

# losowanie z rozkładu Poissona
h1 = n.random.poisson(5, 100000)
# przygotowanie histogramu z rozkładu Poissona
count1, bins1, ignored1 = pl.hist(h1, 14, density=True,
color='r', label='Rozkład Poissona')
# losowanie z rozkładu trójkątnego
h2 = n.random.triangular(15, 20, 25, 100000)
# przygotowanie histogramu z rozkładu trójkątnego
count2, bins2, ignored2 = pl.hist(h2, 50, density=True,
color='b', label='Rozkład trójkątny')

pl.ylabel('Gęstosc prawdopodobienstwa')
pl.xlabel('argument')
pl.legend()
pl.show()
```

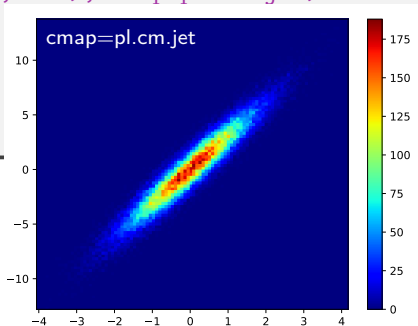
Histogramy 2D

```
import numpy as n
import matplotlib.pyplot as pl

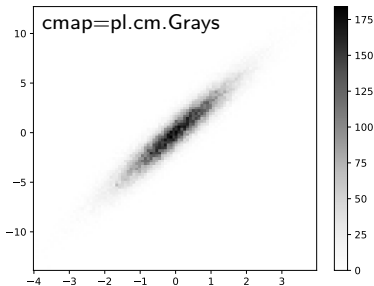
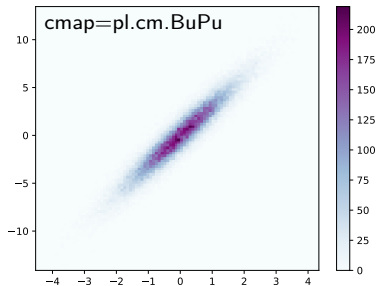
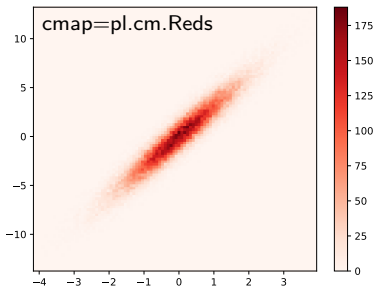
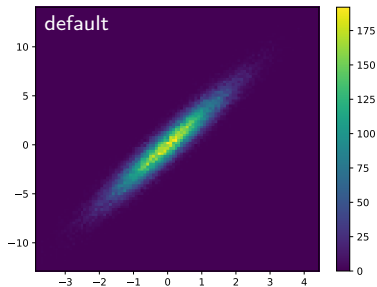
# create data
x = n.random.normal(size=50000)
y = x * 3 + n.random.normal(size=50000)

# construct histogram 2D: data, nbins, color_map
pl.hist2d(x, y, bins=(100, 100), cmap=pl.cm.jet)
# show z-scale
pl.colorbar()

pl.show()
```



Histogramy 2D - palety kolorów



Wykresy 3D

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
import pylab as p
import numpy as np

# wytwarzamy obiekt typu figura z osiami 3D
fig = p.figure()
ax = Axes3D(fig)

# tworzymy wektory opisujące osie X i Y
X = np.arange(-15, 15, 0.3)
Y = np.arange(-15, 15, 0.3)

# zamieniamy wektory w siatkę, której węzły są zadane przez elementy w macierzach X i Y
X, Y = np.meshgrid(X, Y)

# dla każdego węzła obliczamy jego odległość od początku układu współrzędnych
R = np.sqrt(X**2 + Y**2)

# dla każdego węzła obliczamy wartość funkcji Z
Z = np.sin(R)/R

# generujemy rysunek powierzchni, rysując np. co drugi węzeł (parametry rstride=2, cstride=2)
ax.plot_surface(X,Y,Z,rstride=1,cstride=1,cmap=cm.jet)
p.show()
```

