

Wstęp do programowania w środowisku PYTHON



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

Kontakt: mkuich@fuw.edu.pl

Materiały: www.fuw.edu.pl/~mkuich/tik2020/

Cechy Python'a

- UWAGA! Korzystamy z Python3
- 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
- Posiada dynamiczny system typów zmiennych
- Najpopularniejsze IDE:
 - ▶ Jupyter, www.jupyter.org
 - ▶ Atom, www.atom.io
 - ▶ Spyder, www.spyder-ide.org
 - ▶ Sublime Text 3, www.sublimetext.com/3
 - ▶ IDLE - zintegrowany
- Istnieje wiele web-owych interpreterów Python'a, np.:
www.programiz.com/python-programming/online-compiler/
- Na pracowni Python3.6 i edytor nano

Uruchamianie

- W zwykłym terminalu wpisujemy `python3`
 - ▶ otworzyliśmy interpreter, w którym mamy dostęp do pełnej funkcjonalności Python'a i interaktywnie możemy pracować. Wypróbuj np.:
 - `print('Hello world')` - polecenie wypisze na ekran ciąg znaków
 - `print('2+3')` - nadal ciąg znaków, bo `' '`, ale
 - `print(2+3)` - wykona operacje matematyczną
 - ▶ 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

1 Pakiet math

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

2 Pakiet numpy

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

3 Pakiet matplotlib

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

4 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

- Działanie w trybie interaktywnym bywa uciążliwe, dlatego najczęściej tworzy się skrypty
- Skrypt to plik tekstowy o rozszerzeniu `.py`, zawierający kod źródłowy programu interpretowanego przez interpreter
- Uruchamianie skryptu w terminalu przez interpreter:
`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`
- Skrypty edytujemy w dowolnym edytorze tekstu lub IDE
- Przykładowy skrypt:

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

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: liczby parzyste od 0 do 10

```
for i in range(0,11,2):  
    print(i)
```

Zadanie 2.

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

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 3.

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.

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)
dict	słownik, czyli tablica asocjacyjna (zmienny)	{'key0': 1.0, 'key1': False}
NoneType	obiekt reprezentujący brak wartości, odpowiednik wartości <code>null</code> z innych języków	None

Uporządkowane sekwencje obiektów

Krotki

- ▷ **niezmiennie**
- ▷ 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

- ▷ **zmiennie**
- ▷ 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]`

Rysowanie z kolekcji

- **Uwaga!** W przypadku pracy na komputerach na pracowni przez usługę secure shell bez transmisji tzw. X-ów, należy zmienić domyślny back-end wyświetlania wykresów przed pakiet matplotlib

Na pracowni

```
import matplotlib
matplotlib.use('Agg')

import matplotlib.pyplot as p

#zbiór punktów x
x0=[1,2,3,4,5]
#zbiór punktów y odpowiadający x
y0=[2,4,6,8,10]

p.plot(x0,y0)
p.plot(x0,y0,'ro')
p.savefig('przyklad1.pdf')
```

Lokalnie

```
import matplotlib.pyplot as p

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

p.plot(x0,y0)
p.plot(x0,y0,'ro')
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 4.

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

```
#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

```
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

```
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

```
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

```
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**

```
import numpy as n

tab = n.loadtxt('array.txt')
print(tab)
print(tab[:, 0])
print(tab[:, 1])
print(tab[:, 2])
```

Rysowanie z tablicy - przykład 7.1

Na pracowni

```
import matplotlib
matplotlib.use('Agg')

import matplotlib.pyplot as p

import numpy as n

#zbiór punktów 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.plot(x1,y1,'m ^ ')
p.savefig('przyklad7.pdf')
```

Lokalnie

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

#zbiór punktów 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.show()
```

Zadanie 5.

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 sinusoidalną 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 7.1 - rysowanie z tablicy**
- ▷ **Przykład 7.2 - wykreślanie funkcji (zadanie 5)**
- ▷ Przykład 7.3 - rysowanie danych z pliku bez niepewności
- ▷ Przykład 7.4 - formatowanie wykresu
- ▷ Przykład 7.5 - rysowanie danych z pliku z niepewnościami
- ▷ Przykład 7.6 - legenda
- ▷ Przykład 7.7 - tekst
- ▷ Przykład 7.8 - osie
- ▷ Przykład 7.9 - siatka, opisy i tytuły

```
import matplotlib
matplotlib.use('Agg')
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('przyklad7.pdf')
```


Rysowanie wykresów - kontynuacja przykładów

- ▷ Przykład 7.1 - rysowanie z tablicy
- ▷ Przykład 7.2 - wykreślanie funkcji (zadanie 5)
- ▷ **Przykład 7.3 - rysowanie danych z pliku bez niepewności**
- ▷ Przykład 7.4 - formatowanie wykresu
- ▷ Przykład 7.5 - rysowanie danych z pliku z niepewnościami
- ▷ Przykład 7.6 - legenda
- ▷ Przykład 7.7 - tekst
- ▷ Przykład 7.8 - osie
- ▷ Przykład 7.9 - siatka, opisy i tytuły

```
import matplotlib
matplotlib.use('Agg')
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('przyklad7.pdf')
```

- ▷ Przykład 7.1 - rysowanie z tablicy
- ▷ Przykład 7.2 - wykreślanie funkcji (zadanie 5)
- ▷ Przykład 7.3 - rysowanie danych z pliku bez niepewności
- ▷ **Przykład 7.4 - formatowanie wykresu**
- ▷ Przykład 7.5 - rysowanie danych z pliku z niepewnościami
- ▷ Przykład 7.6 - legenda
- ▷ Przykład 7.7 - tekst
- ▷ Przykład 7.8 - osie
- ▷ Przykład 7.9 - siatka, opisy i tytuły

```
import matplotlib
matplotlib.use('Agg')
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('przyklad7.pdf')
```

- ▷ Przykład 7.1 - rysowanie z tablicy
- ▷ Przykład 7.2 - wykreślanie funkcji (zadanie 5)
- ▷ Przykład 7.3 - rysowanie danych z pliku bez niepewności
- ▷ Przykład 7.4 - formatowanie wykresu
- ▷ **Przykład 7.5 - rysowanie danych z pliku z niepewnościami**
- ▷ Przykład 7.6 - legenda
- ▷ Przykład 7.7 - tekst
- ▷ Przykład 7.8 - osie
- ▷ Przykład 7.9 - siatka, opisy i tytuły

```
import matplotlib
matplotlib.use('Agg')
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('przyklad7.pdf')
```

- ▷ Przykład 7.1 - rysowanie z tablicy
- ▷ Przykład 7.2 - wykreślanie funkcji (zadanie 5)
- ▷ Przykład 7.3 - rysowanie danych z pliku bez niepewności
- ▷ Przykład 7.4 - formatowanie wykresu
- ▷ Przykład 7.5 - rysowanie danych z pliku z niepewnościami
- ▷ **Przykład 7.6 - legenda**
- ▷ Przykład 7.7 - tekst
- ▷ Przykład 7.8 - osie
- ▷ Przykład 7.9 - 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('przyklad7.pdf')
```

Rysowanie wykresów - kontynuacja przykładów

- ▷ Przykład 7.1 - rysowanie z tablicy
- ▷ Przykład 7.2 - wykreślanie funkcji (zadanie 5)
- ▷ Przykład 7.3 - rysowanie danych z pliku bez niepewności
- ▷ Przykład 7.4 - formatowanie wykresu
- ▷ Przykład 7.5 - rysowanie danych z pliku z niepewnościami
- ▷ Przykład 7.6 - legenda
- ▷ **Przykład 7.7 - tekst**
- ▷ Przykład 7.8 - osie
- ▷ Przykład 7.9 - siatka, opisy i tytuły

```
...  
p.legend((11, 12, 13), ('array',  
    'function', 'file'))  
  
p.text(1,4, 'ąęłńóśź')  
p.text(1, 5, r'  
  
    $\Delta=\frac{1}{  
r^2}\frac{\partial}{\partial r}  
r^2\frac{\partial}{\partial r}+  
\frac{1}{r^2}\left(\frac{1}{\sin  
\theta}\frac{\partial}{\partial  
\theta}\sin\theta\frac{\partial}{\partial  
\partial\theta}+\frac{1}{  
\sin^2\theta}\frac{\partial^2}{  
\partial\phi^2}\right)$  
  
' )  
p.savefig('przyklad7.pdf')
```

Rysowanie wykresów - kontynuacja przykładów

- ▷ Przykład 7.1 - rysowanie z tablicy
- ▷ Przykład 7.2 - wykreślanie funkcji (zadanie 5)
- ▷ Przykład 7.3 - rysowanie danych z pliku bez niepewności
- ▷ Przykład 7.4 - formatowanie wykresu
- ▷ Przykład 7.5 - rysowanie danych z pliku z niepewnościami
- ▷ Przykład 7.6 - legenda
- ▷ Przykład 7.7 - tekst
- ▷ **Przykład 7.8 - osie**
- ▷ Przykład 7.9 - siatka, opisy i tytuły

```
...
```

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

```
p.savefig('przyklad7.pdf')
```

→ pierwsza i druga liczba określają zakres osi X

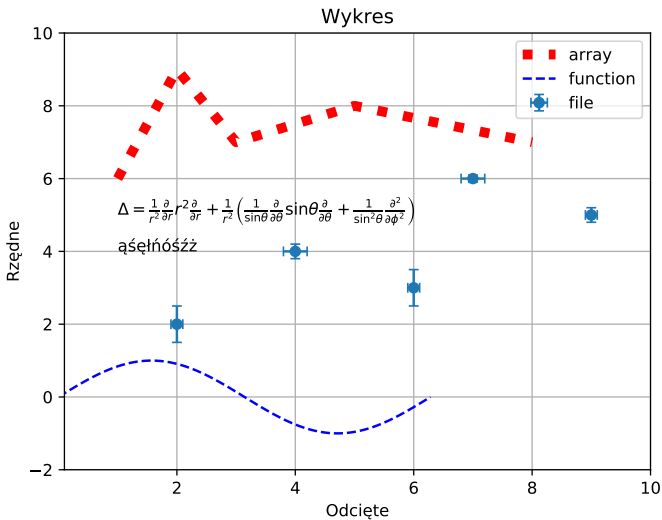
→ kolejne dwie liczby określają zakres osi Y

Rysowanie wykresów - kontynuacja przykładów

- ▷ Przykład 7.1 - rysowanie z tablicy
- ▷ Przykład 7.2 - wykreślanie funkcji (zadanie 5)
- ▷ Przykład 7.3 - rysowanie danych z pliku bez niepewności
- ▷ Przykład 7.4 - formatowanie wykresu
- ▷ Przykład 7.5 - rysowanie danych z pliku z niepewnościami
- ▷ Przykład 7.6 - legenda
- ▷ Przykład 7.7 - tekst
- ▷ Przykład 7.8 - osie
- ▷ **Przykład 7.9 - siatka, opisy i tytuły**

```
...  
  
pl.grid(True)  
  
p.xlabel('Odcięte')  
p.ylabel('Rzędne')  
  
p.title('Wykres')  
  
p.savefig('przyklad7.pdf')
```

“Przykładowy wykres”



Wzmianka o subplot'cie

```
import matplotlib
matplotlib.use('Agg')
import numpy as n
import matplotlib.pyplot as p
def f(t):
    return n.exp(-t) * n.cos(2*n.pi*t)

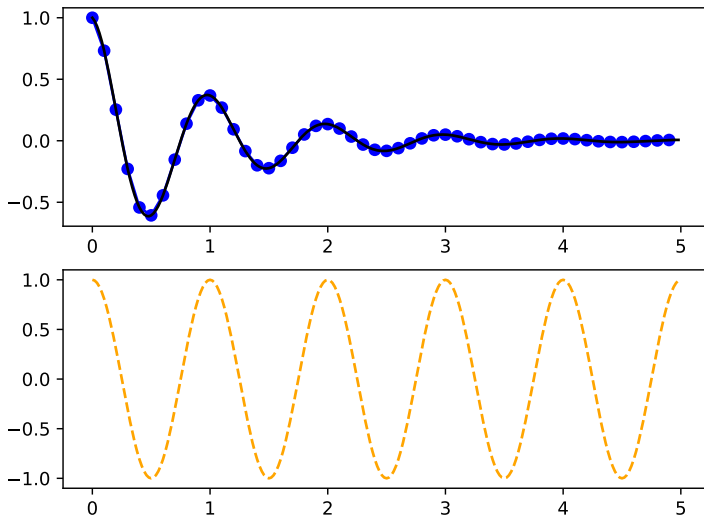
t1 = n.arange(0.0, 5.0, 0.1)
t2 = n.arange(0.0, 5.0, 0.02)

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

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

p.savefig('subprzyklad7.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 8.1 - wykreślanie danych z pliku, bez niepewności**
- ▷ Przykład 8.2 - dopasowywanie prostej oraz drukowanie wartości i błędów parametrów
- ▷ Przykład 8.3 - wykreślanie dopasowanej zależności
- ▷ Przykład 8.4 - zadawanie wartości początkowych parametrów fitu
- ▷ Przykład 8.5 - uwzględnianie błędów na osi rzędnych

```
import matplotlib
matplotlib.use('Agg')

import numpy as n
import matplotlib.pyplot as p

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

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

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

Dopasowanie funkcji do danych - przykłady

- ▷ Przykład 8.1 - wykreślanie danych z pliku, bez niepewności
- ▷ **Przykład 8.2 - dopasowywanie prostej oraz drukowanie wartości i błędów parametrów**
- ▷ Przykład 8.3 - wykreślanie dopasowanej zależności
- ▷ Przykład 8.4 - zadawanie wartości początkowych parametrów fitu
- ▷ Przykład 8.5 - uwzględnianie błędów na osi rzędnych

```
...
from scipy.optimize import curve_fit
data = n.loadtxt('line.txt')
x = data[:, 0]
y = data[:, 1]
p.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)))

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

Dopasowanie funkcji do danych - przykłady

- ▷ Przykład 8.1 - wykreślanie danych z pliku, bez niepewności
- ▷ Przykład 8.2 - dopasowywanie prostej oraz drukowanie wartości i błędów parametrów
- ▷ **Przykład 8.3 - wykreślanie dopasowanej zależności**
- ▷ Przykład 8.4 - zadawanie wartości początkowych parametrów fitu
- ▷ Przykład 8.5 - uwzględnianie błędów na osi rzędnych

```
...  
  
#przygotowanie współrzędnych x z danego zakresu  
xpar = n.arange(-10, 10, 0.1)  
  
#rysowanie dopasowanej zależności  
p.plot(xpar, f(xpar,par[0],par[1]))  
  
#p.plot(xpar, f(xpar, *par))  
  
p.savefig('przyklad8.pdf')
```

Dopasowanie funkcji do danych - przykłady

- ▷ Przykład 8.1 - wykreślanie danych z pliku, bez niepewności
- ▷ Przykład 8.2 - dopasowywanie prostej oraz drukowanie wartości i błędów parametrów
- ▷ Przykład 8.3 - wykreślanie dopasowanej zależności
- ▷ **Przykład 8.4 - zadawanie wartości początkowych parametrów fitu**
- ▷ Przykład 8.5 - 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)  
p.plot(xpar, f(xpar, par[0], par[1]))  
  
p.savefig('przyklad8.pdf')
```


Dopasowanie funkcji do danych - przykłady

- ▷ Przykład 8.1 - wykreślanie danych z pliku, bez niepewności
- ▷ Przykład 8.2 - dopasowywanie prostej oraz drukowanie wartości i błędów parametrów
- ▷ Przykład 8.3 - wykreślanie dopasowanej zależności
- ▷ Przykład 8.4 - zadawanie wartości początkowych parametrów fitu
- ▷ **Przykład 8.5 - uwzględnianie błędów na osi rzędnych**

```
...  
data = n.loadtxt('line.txt')  
x = data[:, 0]  
y = data[:, 1]  
s = data[:, 2]  
p.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)  
  
...  
  
p.savefig('przyklad8.pdf')
```

Histogramy: Przykład 9.1 - wykreślanie histogramu

```
import matplotlib
matplotlib.use('Agg')

import matplotlib.pyplot as p
import numpy as n

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

p.hist(data)

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

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

Histogramy: Przykład 9.2 - fitowanie histogramu

```
...
data = n.loadtxt('wahadlo.txt')
#p.hist(data)
num, bins, patches = p.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)
p.plot(x,gauss(x, *par), 'g-')
...
p.savefig('przyklad9.pdf')
```

Zadania do samodzielnego rozwiązania na stronie ćwiczeń w dziale python:

www.fuw.edu.pl/~mkuich/tik2020/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 p

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
p.xlim(5,26)
# przygotowanie histogramu z bledami
p.bar(bincenters,y,color='g',width=w,yerr=err)
p.title("Liczby losowe wygenerowane z plaskiego rozkladu")
p.show()
```

Liczby pseudo-losowe i histogramy (2)

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

# losowanie z rozkładu Poissona
h1 = n.random.poisson(5, 100000)
# przygotowanie histogramu z rozkładu Poissona
count1, bins1, ignored1 = p.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 = p.hist(h2, 50, density=True,
color='b', label='Rozkład trójkątny')

p.ylabel('Gęstość prawdopodobieństwa')
p.xlabel('argument')
p.legend()
p.show()
```

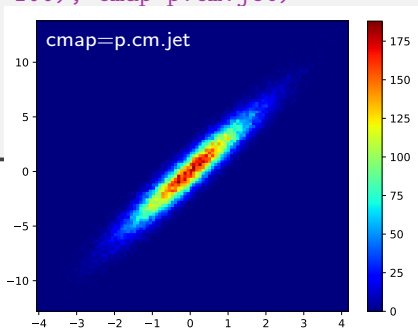
Histogramy 2D

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

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

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

p.show()
```



Histogramy 2D - palety kolorów

