Matematyczne aspekty analizy danych
semestr zimowy 2025/2026

Dr Anna Muranova
UWM w Olsztynie

Wykład 2

Biblioteka `math`

```
import math

print(math.factorial(5))
```

Funkcje: import math

Link do dokumentacji https://docs.python.org/3/library/math.html

- ```
  import math

  a=0
  b=math.sin(2*math.pi)
  print(b)
  print(math.isclose(a,b, rel_tol=1e-09, abs_tol=1e-09))
  ```
- ```
  import math

  print(math.e)
  print(math.exp(1))
  print(math.ceil(math.e))
  print(math.floor(math.e))
  print(math.factorial(10))
  ```

Funkcje: import math

Link do dokumentacji `https://docs.python.org/3/library/math.html`

- ```
  import math

  print(math.gcd())
  print(math.gcd(20,16))
  print(math.gcd(20,15,35,25))
  ```
- ```
  import math

  #print(math.prod())
  #print(math.prod(0,2))
  print(math.prod([0,2]))
  print(math.prod([12,3,21]))
  print(math.prod([12,3,21], start=2))
  ```

Funkcje: logarytm

Link do dokumentacji `https://docs.python.org/3/library/math.html`

- math.log(x, base)

```
import math

# Return the natural logarithm of different numbers
print(math.log(2.7183))
print(math.log(2))
print(math.log(1))
```

- `import math`

```
print(math.log(100,10))
print(math.log(1,10))
print(math.log(0.25,2))
```

Funkcje: potęga

```
import math

print(math.pow(2,100))
print(pow(2,100))
print(2**100)

#print(math.pow(-2,0.5))
print(pow((-2),0.5))
print((-2)**0.5)

print(math.exp(1e-5) - 1)  # gives result accurate to 11 places
print(math.expm1(1e-5) )
```

Oprócz tego `math.exp(x)` dokładniej niż `math.e ** x` oraz `pow(math.e, x)`.

Funkcje: reszta

```
import math

print(math.remainder(8,3))
print(math.fmod(8,3))
print(8%3)

print(math.remainder(8.5,3))
print(math.fmod(8.5,3))
print(8.5%3)

print(math.remainder(-1e-100,1e100))
print(math.fmod(-1e-100,1e100))
print(-1e-100 % 1e100)
```

Funkcje

Funkcje – to wyrażenia, które definiują relacje miedzy dwiema lub wieloma zmiennymi: mówiąc ścisłej, funkcja przyjmuje *zmienny wejściowe*, umieszcza je w wyrażeniu i generuje *zmienny wyjściową*.
Na przykład,

$$y = 2x + 1 \text{ lub } f(x) = 2x + 1$$

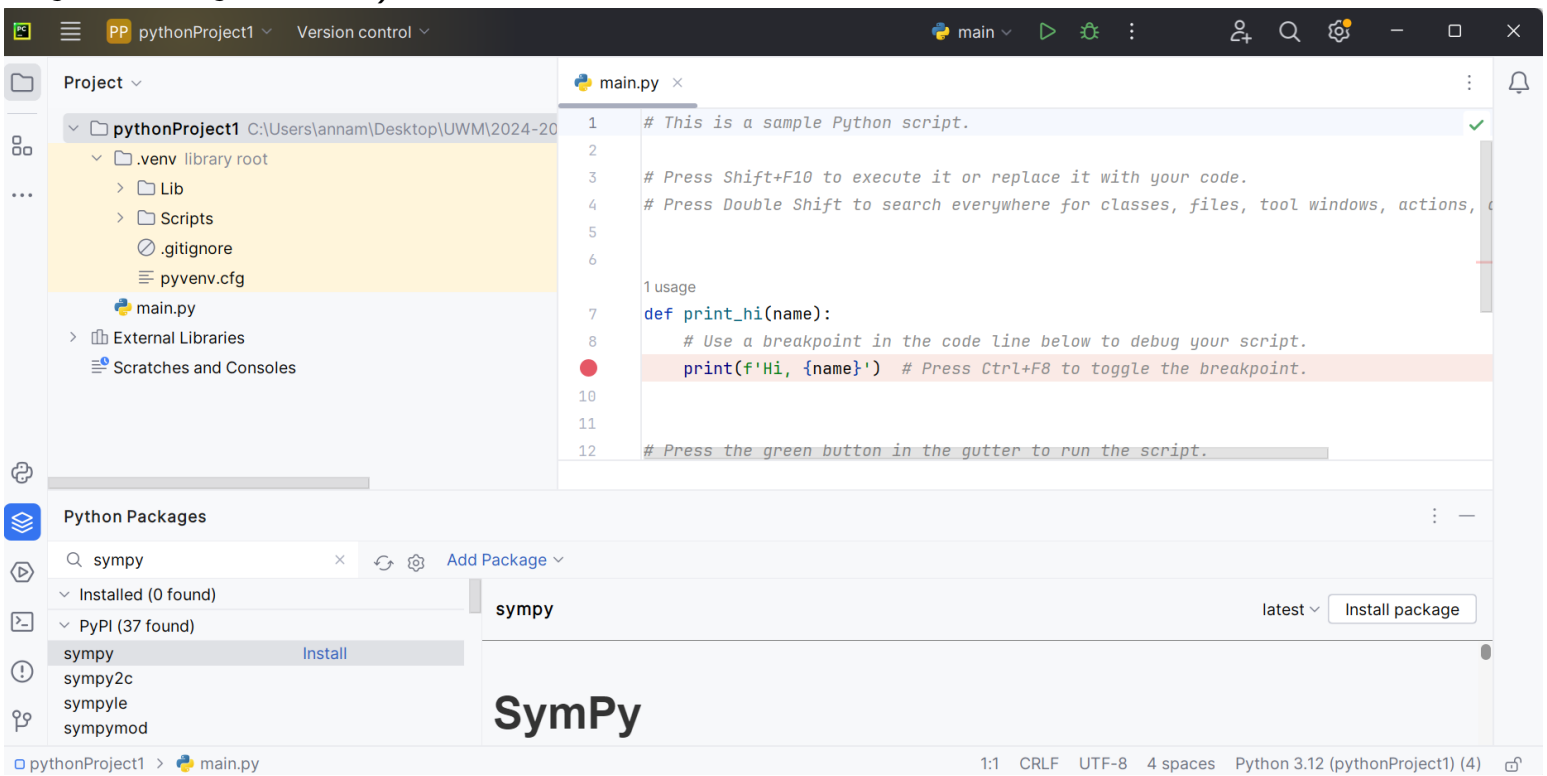
x	y = f(x)
0	1
0.5	2
1.25	3.5
3	7

```
def f(x):
    return 2 * x + 1


x_values = [0, 0.5, 1.25, 3]
for x in x_values:
    print(f(x))
```

Rozdział 2. Podstawy matematyki oraz pakiet `sympy`

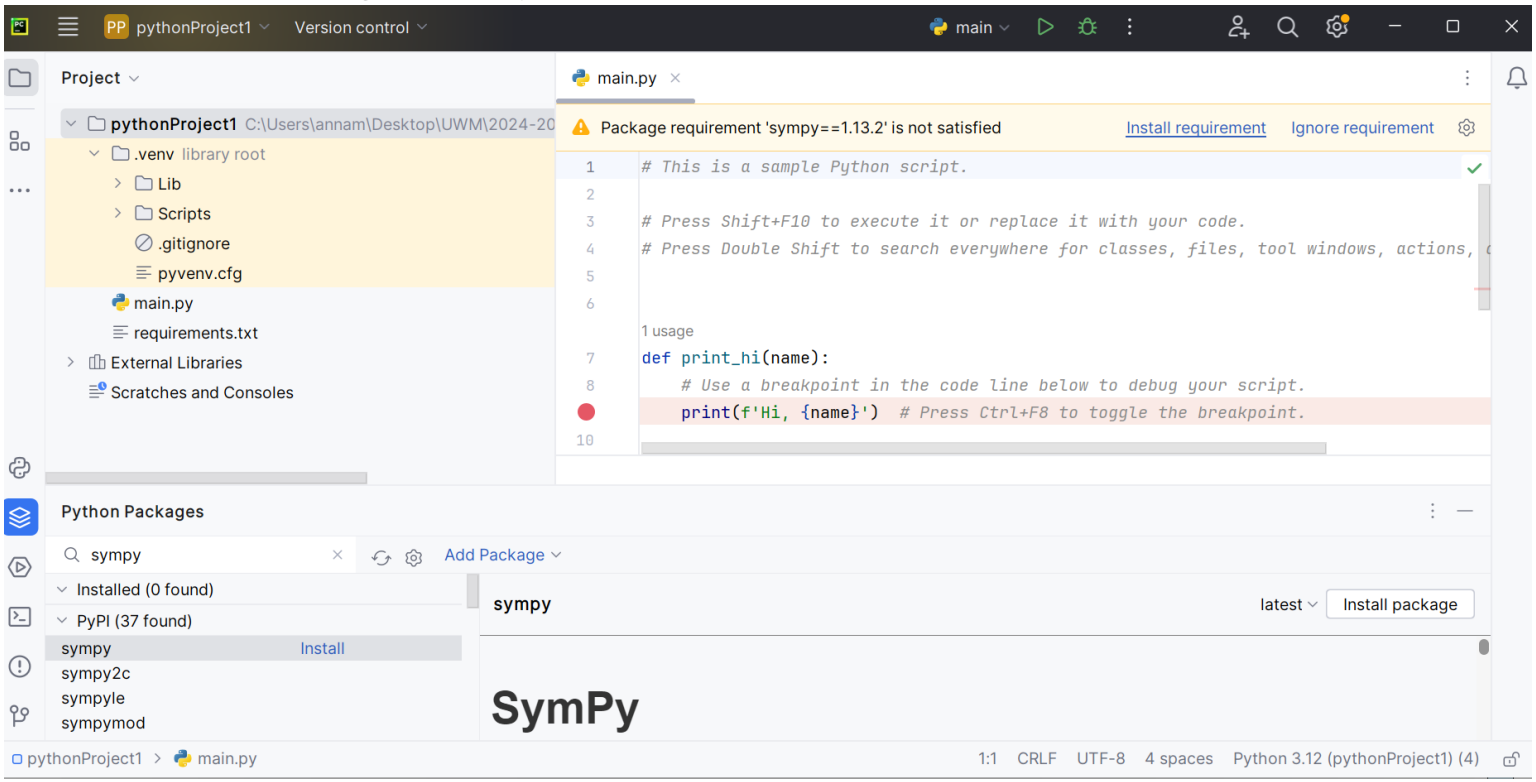
Instalowanie pakietów

1 sposób
Python packages –> Wyszukać pakiet **sympy** i zainstalować (lepiej w najnowszej wersie)



Instalowanie pakietów

2 sposób
Dodać plik `requirements.txt`:
`http://wmii.uwm.edu.pl/~muranova/MAAD2024-25/requirements.txt`
do folderu projektu w przeglądarce plików, otworzyć ten plik w projekcie w **PyCharmie** i na gorze wybrac *Install requirements*



Instalowanie pakietów

2 sposób
Dlaczego może nie być tego banneru na gorze:

- `Reader mode` musi być wyłączony:
  File –> Settings –> Editor –> Reader Mode
- Settings –> Editor –> Inspections –> unsatisfied package requirements musi być zaznaczone

Więcej na ten temat:
`https://www.jetbrains.com/help/pycharm/managing-project-dependencies.html`

Biblioteka Sympy

SymPy jest biblioteką matematyczną (systemem algebry komputerowej lub komputerowym systemem obliczeń symbolicznych), która została napisana od podstaw w języku Python i używa go jako narzędzia interakcji (nie trzeba uczyć się nowego języka programowania, jak w przypadku większości innych systemów tego typu).

Co to jest matematyka symboliczna?

Python operuje na liczbach zmiennoprzecinkowych:

```
import math
a = math.sqrt(3)
print(a)
print(a**2)
```

W matematyce symbolicznej operujemy na liczbach i symbolach

```
import sympy
a = sympy.sqrt(3)
print(a)
print(a.evalf())
print(a.evalf(50))
print(a**2)
```

Jeszcze przykład

```
import math
import sympy

print(math.sin(math.pi))
print(math.pi)
print(sympy.sin(sympy.pi))
print(sympy.pi)

print(math.log(math.e))
print(math.e)
print(sympy.log(sympy.E))
print(sympy.E)
```

Uwaga: inny sposób podłączania pakietu

```
from sympy import *
from math import *
print(E)
print(e)
print(E==e)
print(E-e)
print(E.evalf()-e)
print(E.evalf()==e)
print(E.evalf())

print(pi)#uwaga!
```

Porównaj:

```
from math import *
from sympy import *

print(pi)
```

Sumowanie w Sympy

```
from sympy import *


i, n = symbols('i n')

# iterujemy po elementach i od 1 do n
# nastepne mnozimy i sumujemy
suma = Sum(2*i,(i,1,n))

#ustawiamy n na 5
#iterujemy po liczbach od 1 do 5
suma_do_5 = suma.subs(n,5)
print(suma_do_5)
print(suma_do_5.doit())
```

Porównaj

```
from sympy import *

print(sum(pow(i,0.5)**2 for i in range(0,11)))


i, n = symbols('i n')
suma = Sum(sqrt(i)**2,(i,1,n))

suma_do_10 = suma.subs(n,10)
print(suma_do_10)
print(suma_do_10.doit())
```

doit() vs evalf

```
from sympy import *

print(sum(pow(i,0.5) for i in range(0,11)))


i, n = symbols('i n')
suma = Sum(sqrt(i),(i,1,n))

suma_do_10 = suma.subs(n,10)
print(suma_do_10)
print(suma_do_10.doit())
print(suma_do_10.evalf())
```

Potęgowanie

$$x^n = \underbrace{x \cdot x \cdot \dots \cdot x}_{n \text{ razy}}$$

Przykład: $x^2 x^3 = x^{2+3} = 5$

$$x^{-n} = 1/x^n$$

Przykład: $x^2/x^5 = x^{2-5} = -3$

$$x^{\frac{p}{q}} = \sqrt[q]{x^p}, x > 0$$

Uwaga:

$$(-4)^{\frac{1}{2}} = \sqrt[2]{-4} = \pm 2i$$

$$(-4)^{\frac{1}{2}} = (-4)^{\frac{2}{4}} = 16^{\frac{1}{4}} = \sqrt[4]{16} = 2$$

Uwaga: $x^0 = 1$, ponieważ

$$1 = x^3/x^3 = x^{3-3} = x^0 \text{ dla } x \neq 0.$$

Zakładamy, że $0^0 = 1$ też.

Potęgi rzeczywiste: $8^{\pi} \approx 8^{\frac{31415926535}{10000000000}} \approx 687.2913$

Potęgowanie w sympy

```
x = symbols ('x')
wyrazenie1 = x**2/x**5
print(wyrazenie1)

wyrazenie2 = x**(1/2)/x**(3/2)
print(wyrazenie2)

wyrazenie3 = x**(Rational(1,2))/x**(Rational(3,2))
print(wyrazenie3)
```

Potęgowanie w math

```
import math

print(math.pow(2,100))
print(pow(2,100))
print(2**100)

#print(math.pow(-2,0.5))
print(pow((-2),0.5))
print((-2)**0.5)
```

Logarytmy

$$a^x = b$$

$$x = \log_a b$$

Do jakiej potęgi podnieść a, żeby otrzymać b?
Uwaga: $a, b > 0, a \neq 1$.

Przykład: $x = \log_2 8 \Rightarrow 2^x = 8 \Rightarrow x = 3$.

Właściwości:

Działanie	Właściwości potęgi	Właściwości logarytmy
Mnożenie	$c^x \cdot c^y = c^{x+y}$	$\log_c(ab) = \log_c(a) + \log_c(b)$
Dzielenie	$\frac{c^x}{c^y} = c^{x-y}$	$\log_c\left(\frac{a}{b}\right) = \log_c(a) - \log_c(b)$
Potęgowanie	$(c^x)^z = c^{xz}$	$\log_c(a^z) = z \log_c a$
Wykładnik zerowy	$c^0 = 1$	$\log_c 1 = 0$

Wskazówka: $x = \log_c a, y = \log_c b$

Logarytmy w math i w sympy

```
import math
import sympy

x = math.log(8,2)
print(x)

x = sympy.log(8,2)
print(x)

x = math.log(10,2)
print(x)
print(2**x)

x = sympy.log(10,2)
print(x)
print(x.evalf())
print(2**x)
print((2**x).evalf())
```

Liczba Eulera

Podstawa logarytmu naturalnego, liczba e, liczba Eulera – stała matematyczna wykorzystywana w wielu dziedzinach matematyki i fizyki.
W przybliżeniu wynosi

$$e \approx 2,718281828459$$

```
import math
import sympy



print(math.e)
print(sympy.E)
print(sympy.E.evalf())

print(math.log(math.e**2))
print(sympy.log(sympy.E**2))
```

Liczba Eulera

Podstawa logarytmu naturalnego, liczba e, liczba Eulera – stała matematyczna wykorzystywana w wielu dziedzinach matematyki i fizyki.
W przybliżeniu wynosi

$$e \approx 2,718281828459$$

```
import math
import sympy



print(math.e)
print(sympy.E)
print(sympy.E.evalf())

print(math.log(math.e**2))
print(sympy.log(sympy.E**2))
```

Definicja 1

Liczba e może być zdefiniowana na kilka równoważnych sposobów.

- Jako granica ciągu, e jest określana przez

$$e = \lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n.$$

```
from sympy import *

n = symbols('n')
print(Limit((1+1/n)**n, n, +oo).doit())

print([(1+1/(10*n))**(10*n) for n in range(1,200)])
```

Definicja 2

Liczba e może być zdefiniowana na kilka równoważnych sposobów.

- Jako suma szeregu e jest określana przez

$$e = \sum_{k=0}^{\infty} \frac{1}{k!} = \lim_{n \to \infty} \sum_{k=0}^{\infty} \frac{1}{k!}$$

```
from sympy import *

k, n = symbols('k n')

suma = Sum(1/factorial(k),(k,0,n))
print(suma)

print(Limit(suma, n, +oo).doit())

suma_do_10 = suma.subs(n,10)#ustawiamy n na 10
print(suma_do_10)
print(suma_do_10.doit())
print(suma_do_10.evalf())
print(suma.subs(n, +oo).doit())
```

Definicja 3

- Za pomocą całki: liczbę e można także zdefiniować jako jedyną liczbę rzeczywistą taką że

$$\int_1^e \frac{1}{t}\,\mathrm{d}t = 1$$

(to znaczy, że liczba e to taka, że pole powierzchni pod hiperbolą $f(t) = 1/t$ od 1 do e jest równe 1).

- Za pomocą funkcji: liczbę e można również zdefiniować jako taki argument funkcji

$$f(x) = x^{1/x}, \quad x > 0$$

dla którego jej wartość jest największa.

- Za pomocą styczną: e jest podstawą takiej funkcji wykładniczej, że styczna do jej wykresu w punkcie $(0, 1)$ ma współczynnik kierunkowy równy 1.

Własności

- pochodna funkcji $(e^x)' = e^x$
- całka funkcji $\int e^x \, \mathrm{d}x = e^x + C$, gdzie C jest dowolną stałą całkowania.
- Jest jednym z elementów wzoru Eulera (zwanego też „najpiękniejszym wzorem matematyki"), wiążącej e z innymi słynnymi liczbami: jednostką urojoną i, π, jednością i zerem:

$$e^{i\pi} = -1$$

(Definicja) e jest podstawą logarytmu naturalnego:

$$\ln x := \log_e x.$$

Funkcje

Funkcje – to wyrażenia, które definiują relacje miedzy dwiema lub wieloma zmiennymi: mówiąc ścisłej, funkcja przyjmuje *zmienny wejściowe*, umieszcza je w wyrażeniu i generuje *zmienny wyjściową*.
Na przykład,

$$y = 2x + 1 \text{ lub } f(x) = 2x + 1$$

x	y = f(x)
0	1
0.5	2
1.25	3.5
3	7

```
def f(x):
    return 2 * x + 1


x_values = [0, 0.5, 1.25, 3]
for x in x_values:
    print(f(x))
```

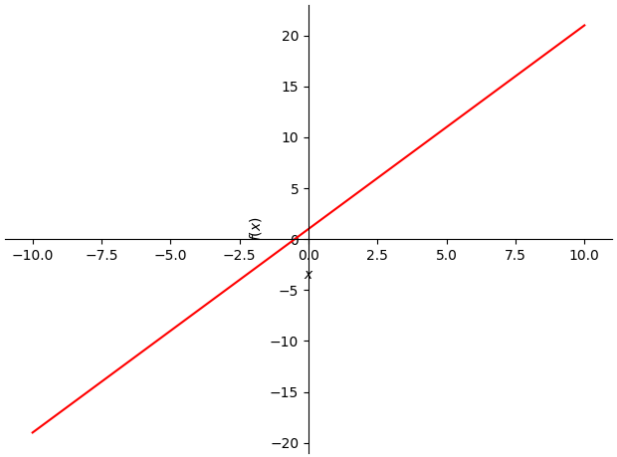
Definiowanie funkcje matematycznych w sympy

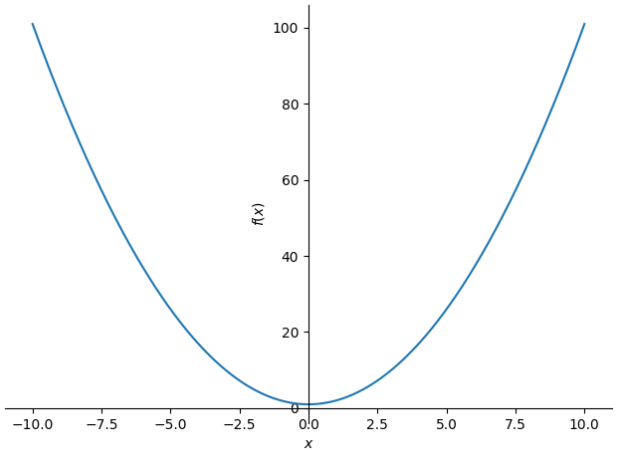
```
from sympy import *

x = symbols('x')
f = 2*x+1

print(f.subs(x, 0.5))
print([f.subs(x, i) for i in  [0, 0.5, 1.25, 3]])
```

Oprócz tego pozwala na działania na funkcjach:

```
from sympy import *

x = symbols('x')
f = x**2-1

print(f.factor())
print(diff(f,x))

print(Limit((1+1/x)**x, x, +oo).doit())
```

Wykresy



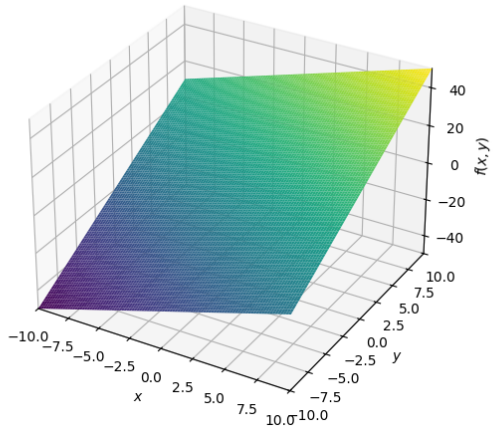


Kody:

```
from sympy import *
x = symbols('x')

f = 2*x+1
plot(f, line_color='red')

plot (x**2 +1)
```

Uwaga: możliwe, że trzeba zainstalować `matplotlib`

Wykresy 3d



Kod:

```
from sympy import *
from sympy.plotting import plot3d

x, y = symbols('x y')

f = 2*x + 3*y
plot3d(f)
```

Uwaga: możliwe, że trzeba zainstalować `matplotlib`