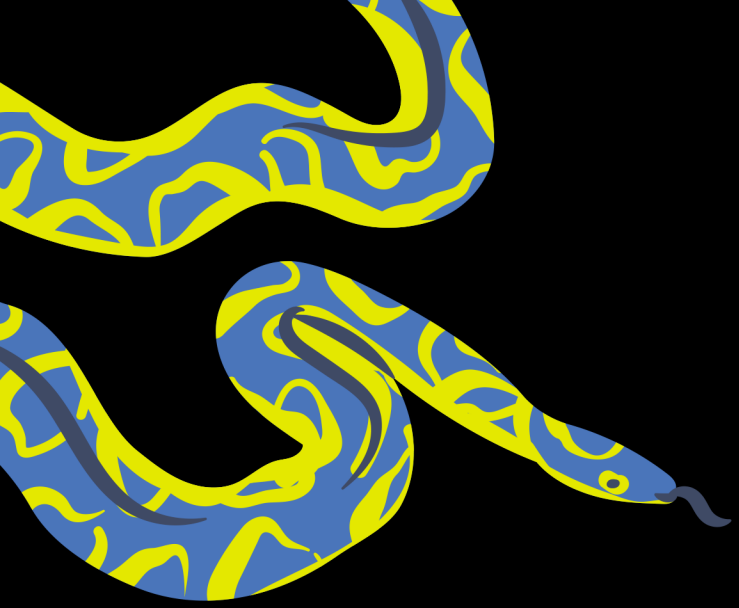
SciByteHub



Python

23.04

SciByteHub

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

```
git config --global user.name “twojanazwa”
```

```
git clone nazwa_repo
```

SciByteHub

Komendy

git init	inicjalizuje nowe repozytorium
git clone	kopiuje istniejące repozytorium
git pull	pobiera zmiany
git push	wysyła zmiany
git add *nazwa*	dodaje plik
git commit -a	zatwierdza wszystkie zmiany
git commit -m “*msg*”	zatwierdza zmiany z wiadomością bez edytora
git status	wyświetla stan repozytorium
git remote add nazwa url	dodaje nowe repozytorium

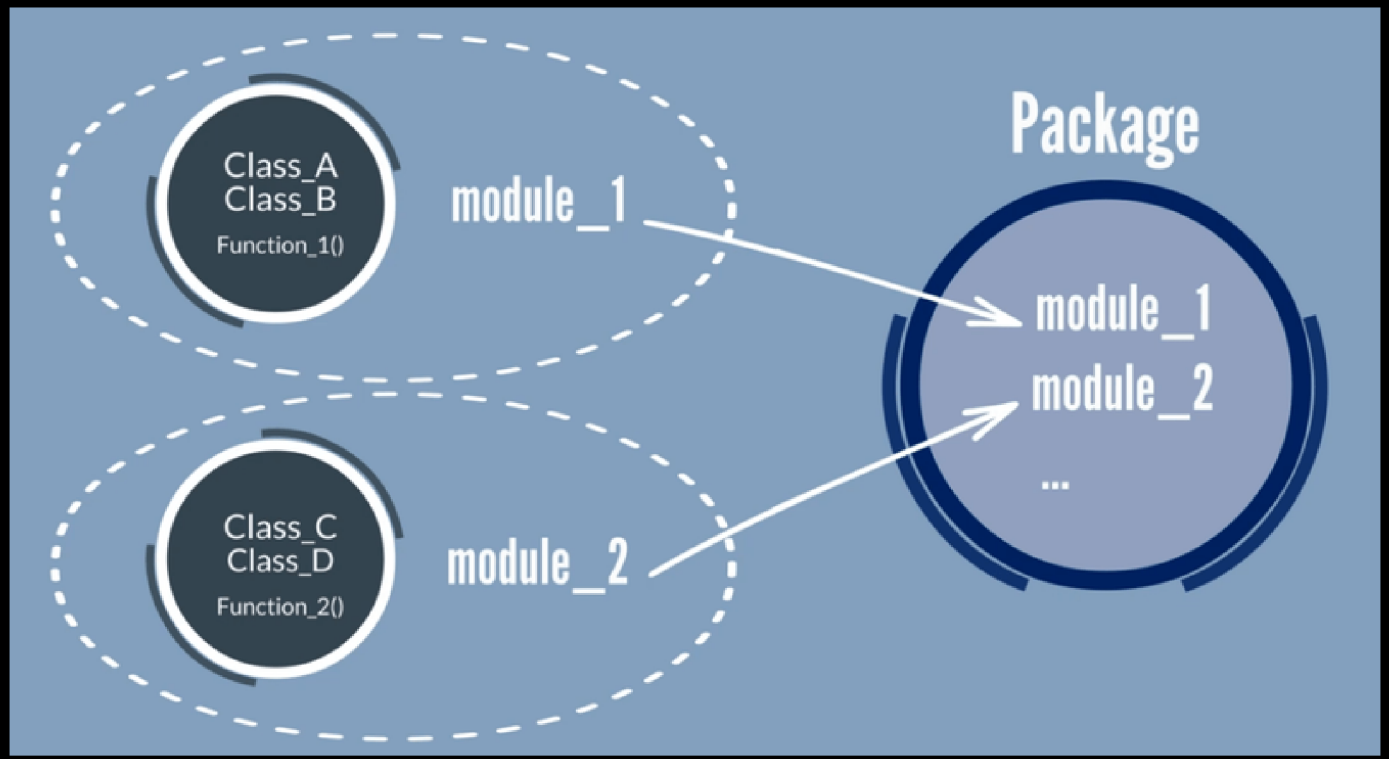
https://github.com/SciByteHub/Python-Kurs-ST-24

SciByteHub

Moduł

↓

kolekcja powiązanych kodów spakowanych w ramach programu Python
pliki Pythona zawierające prawidłowe definicje i instrukcje Pythona z przyrostkiem .py



Functions, Modules and Packages

SciByteHub

Wbudowane	Zdefiniowane
np. math, random	stworzone przez użytkownika

Moduł to plik zawierający definicje i instrukcje Pythona.
Nazwą pliku jest nazwa modułu z dodanym sufiksem .py.

SciByteHub

Wprowadzenie do importu bibliotek i modułów

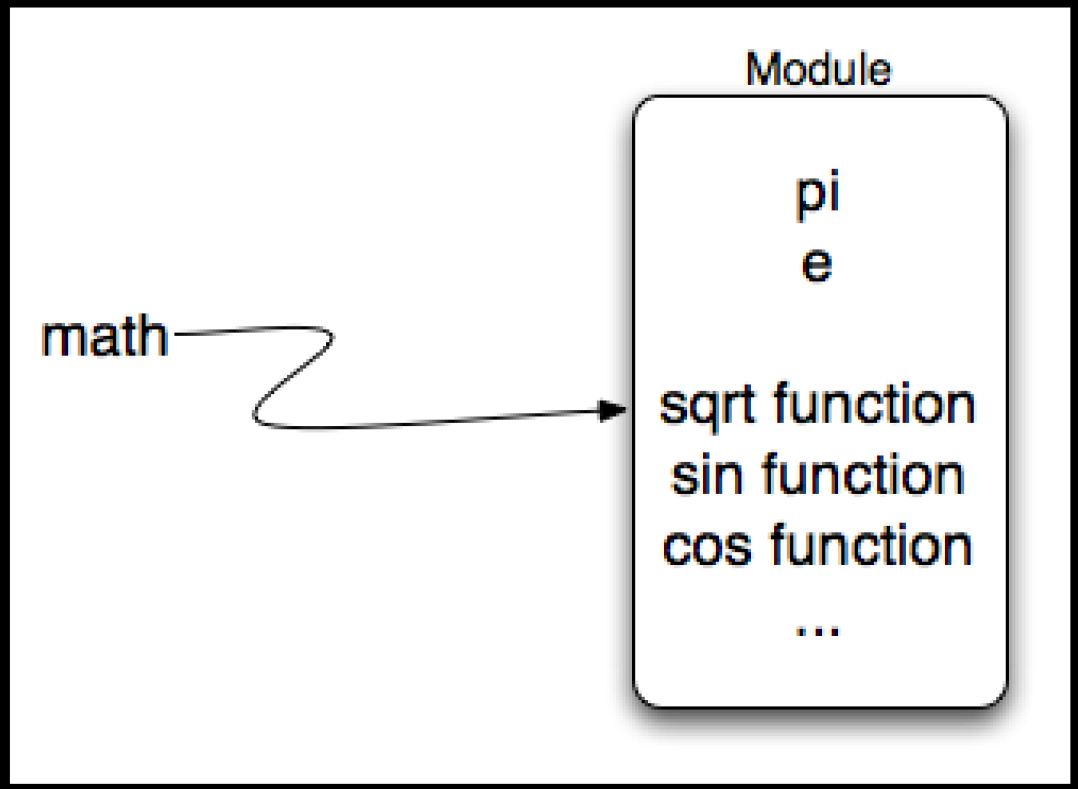
pip install SomePackage

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
from statistics import mean
```

SciByteHub

Moduł math

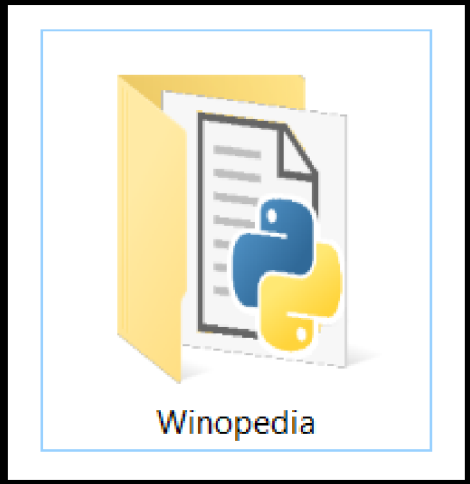
acos(x) - arccos
asin(x) - arcsin
log10(x) - logarytm dziesiętny
pi - stała pi
e - stała e



Math module

Moduł Winopedia jako przykład modułu stworzonego przez użytkownika:

SciByteHub



Nazwa	Data modyfikacji	Typ	Rozmiar
food_pairing	12.12.2023 09:31	Folder plików	
red_wines	12.12.2023 09:31	Folder plików	
white_wines	12.12.2023 09:31	Folder plików	
__init__	12.12.2023 09:09	Python File	1 KB

Ten komputer > Pulpit > Nowy folder (2) > Importy > Winopedia > red_wines

Nazwa	Data modyfikacji	Typ
__init__	12.12.2023 09:06	Python File
cabernet_sauvignon	12.12.2023 08:26	Python File
merlot	12.12.2023 08:26	Python File
pinot_noir	12.12.2023 08:26	Python File

EXPLORER

- WINOPEDIA
  - food_pairing
    - __init__.py
    - pairing.py
  - red_wines
    - __init__.py
    - cabernet_sauvignon.py
    - merlot.py
    - pinot_noir.py
  - white_wines
  - __init__.py

cabernet_sauvignon.py

red_wines > cabernet_sauvignon.py

```python
def characteristics():
    return {
        "struktura": "pełna",
        "smaki": ["czarna porzeczka", "cedr", "tabaka", "czekolada"],
        "najlepsze regiony": ["Bordeaux we Francji", "Dolina Napa w USA", "Coonawarra w Australii"],
        "temperatura podawania": "16-18°C",
        "potencjał starzenia": "10-20 lat"
    }

def recommended_food():
    return ["steki", "danie z dziczyzny", "bogate sery"]


```

SciByteHub

```
[8]
import Winopedia
dir(Winopedia)
```

Pytho

```
['__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__path__',
 '__spec__',
 'food_pairing',
 'red_wines',
 'white_wines']
```

```
[9]
from Winopedia.red_wines import cabernet_sauvignon
```

Pytho

```
[10]
characteristics = cabernet_sauvignon.characteristics()
print(characteristics)
```

Pytho

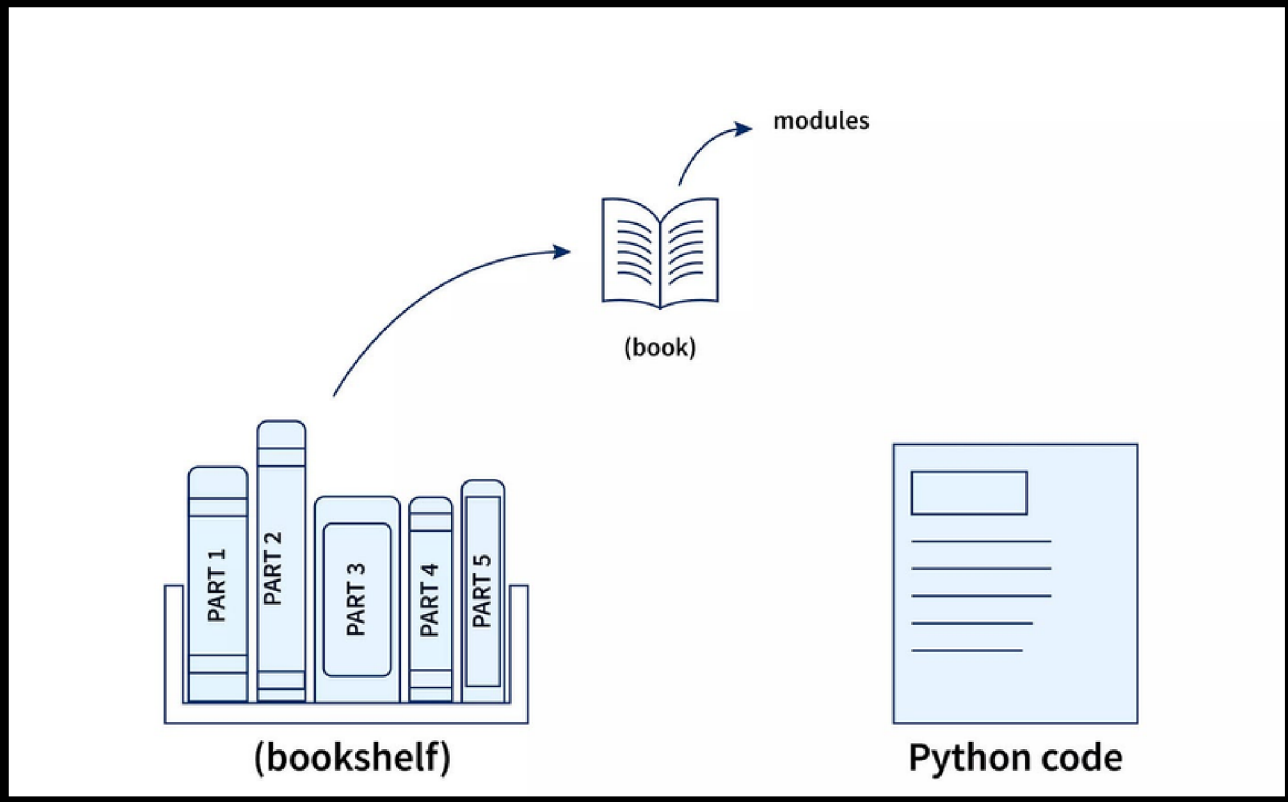
```
{'struktura': 'pełna', 'smaki': ['czarna porzeczka', 'cedr', 'tabaka', 'czekolada'], 'najlepsze regiony': ['Bordeaux we Francji', 'Dolina Napa w USA',
```

SciByteHub

Biblioteki

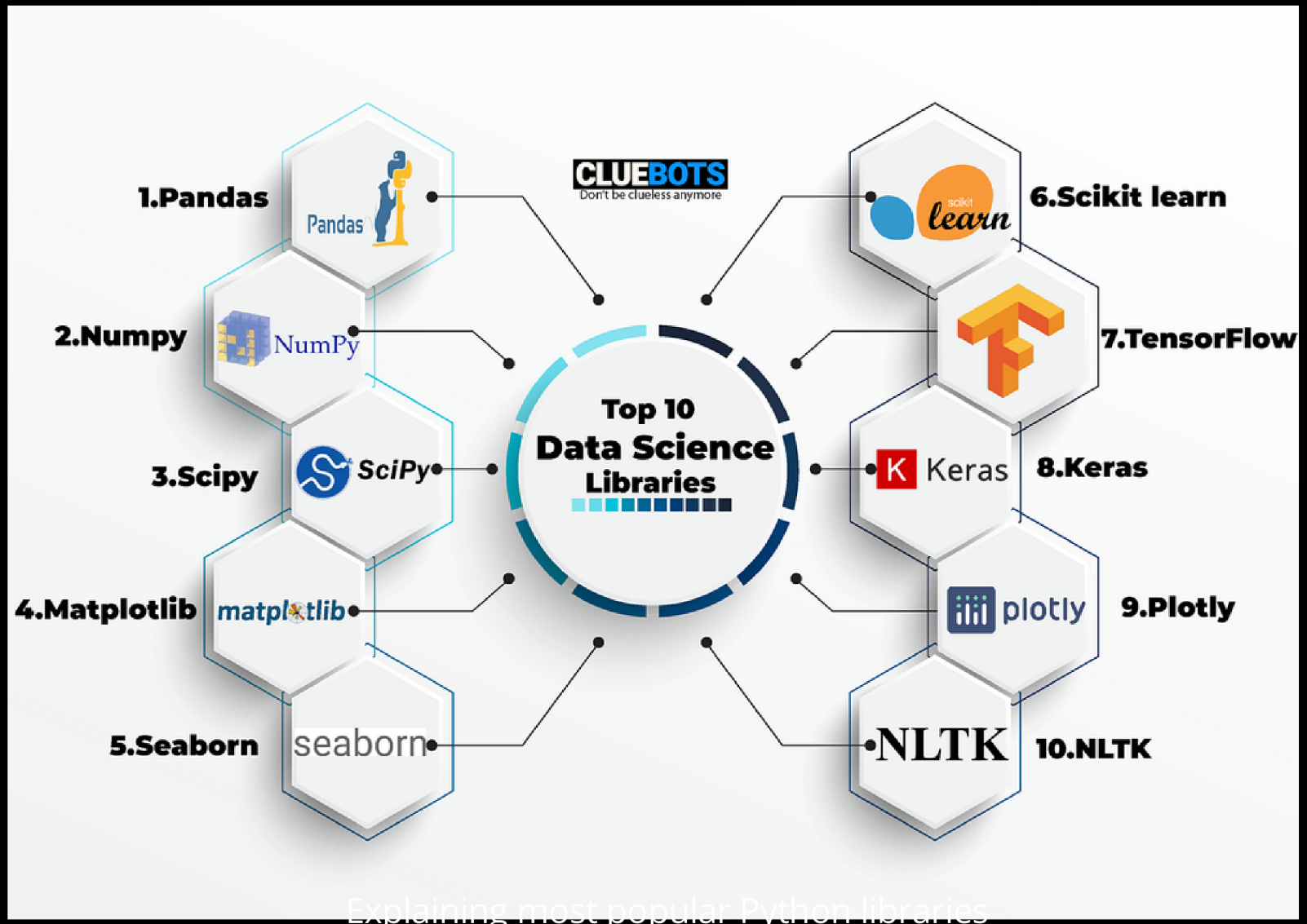
↓

zestaw kodu/instrukcji Pythona wielokrotnego użytku,
najczęściej zbiór powiązanych modułów



Python Libraries

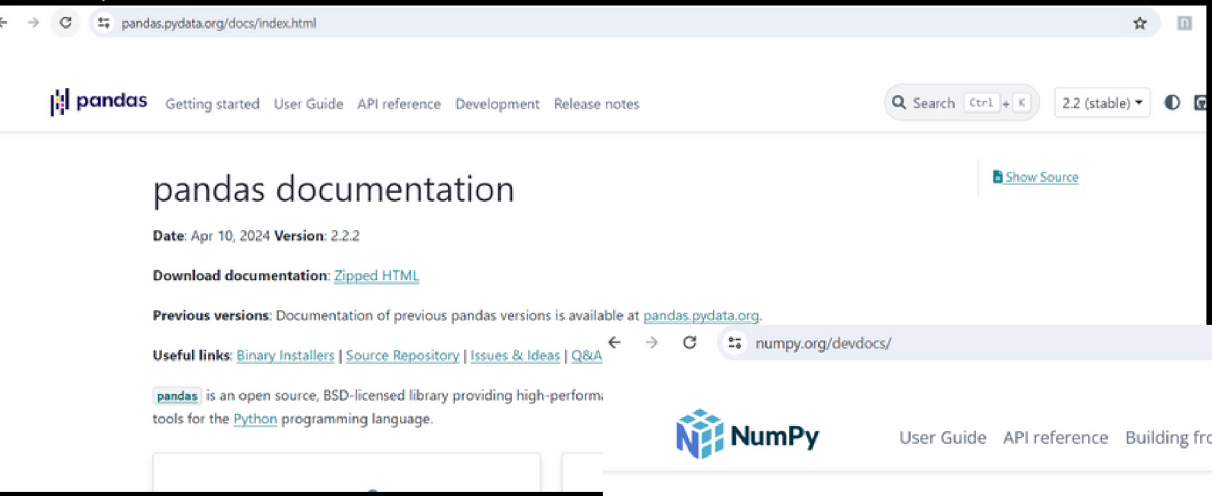
SciByteHub

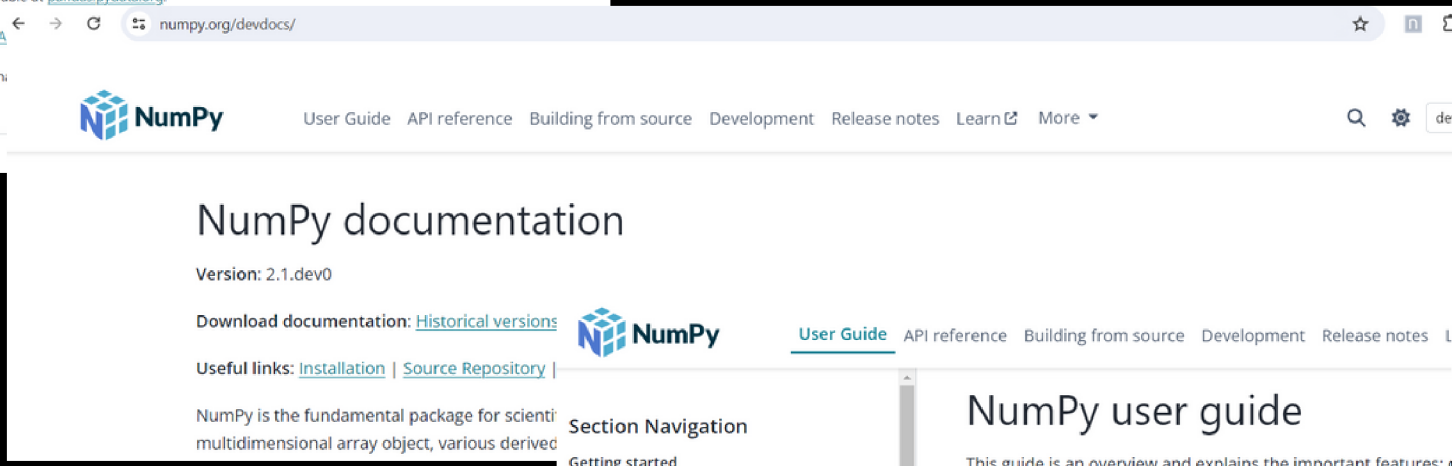


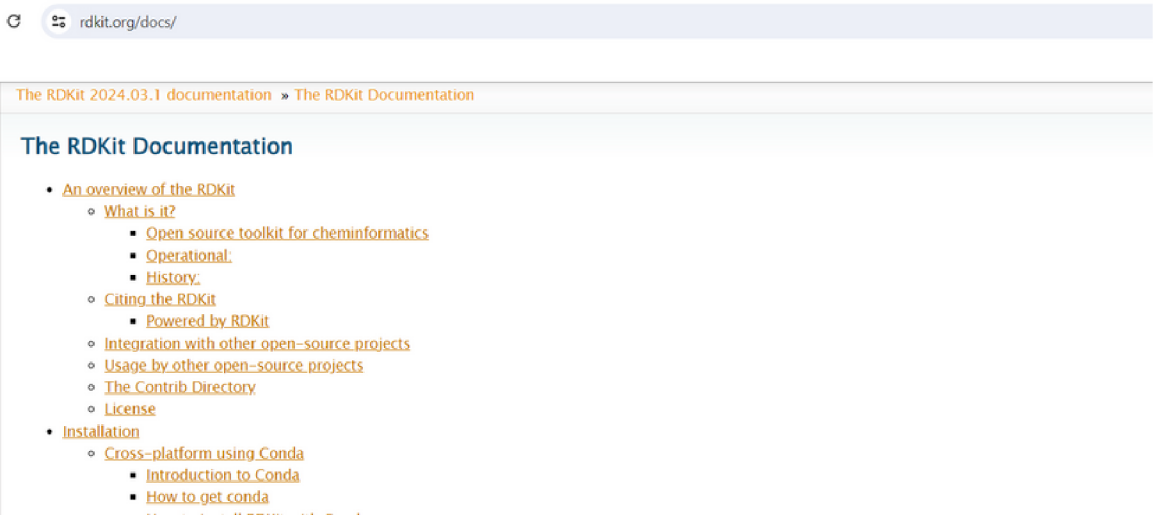
Explaining most popular Python libraries

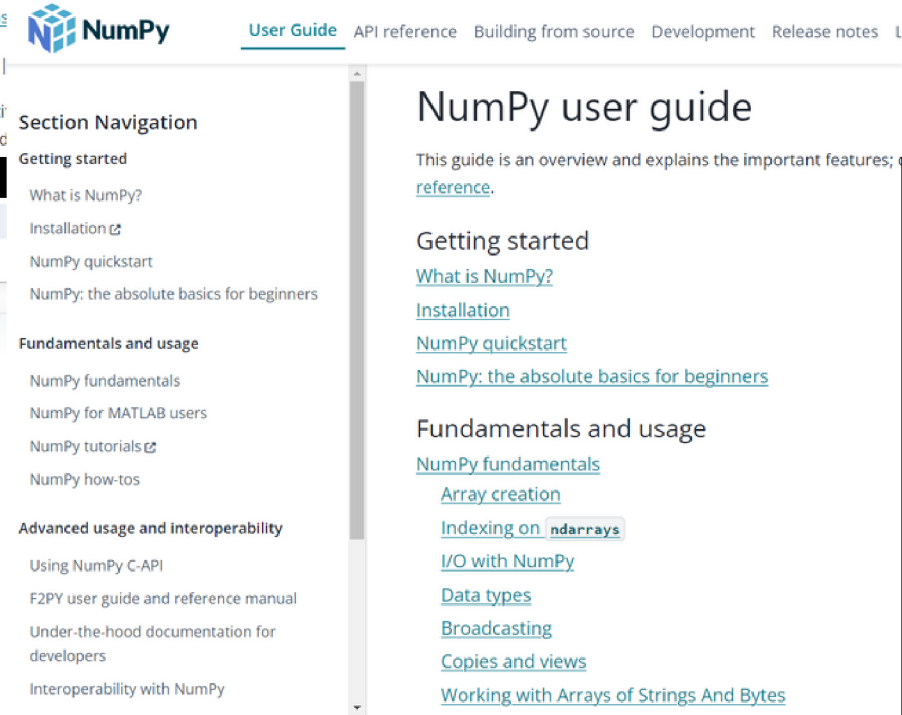
SciByteHub

Dokumentacja







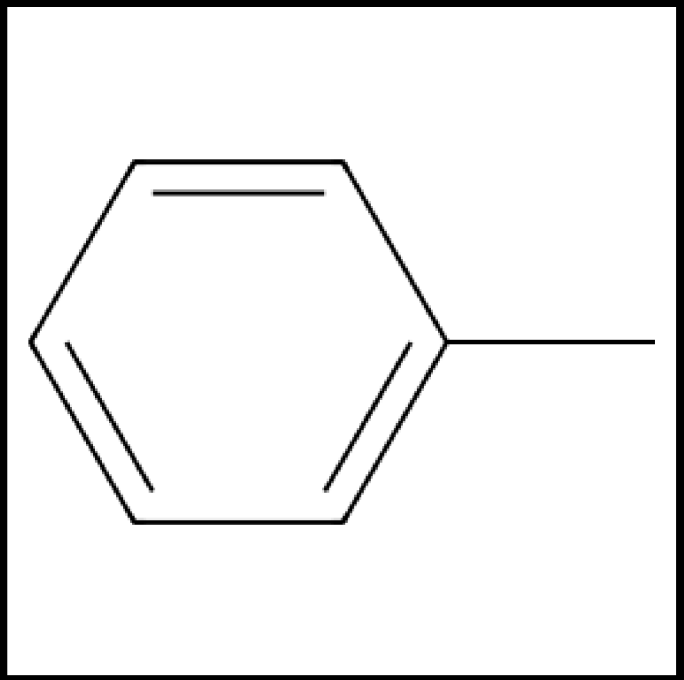


SciByteHub

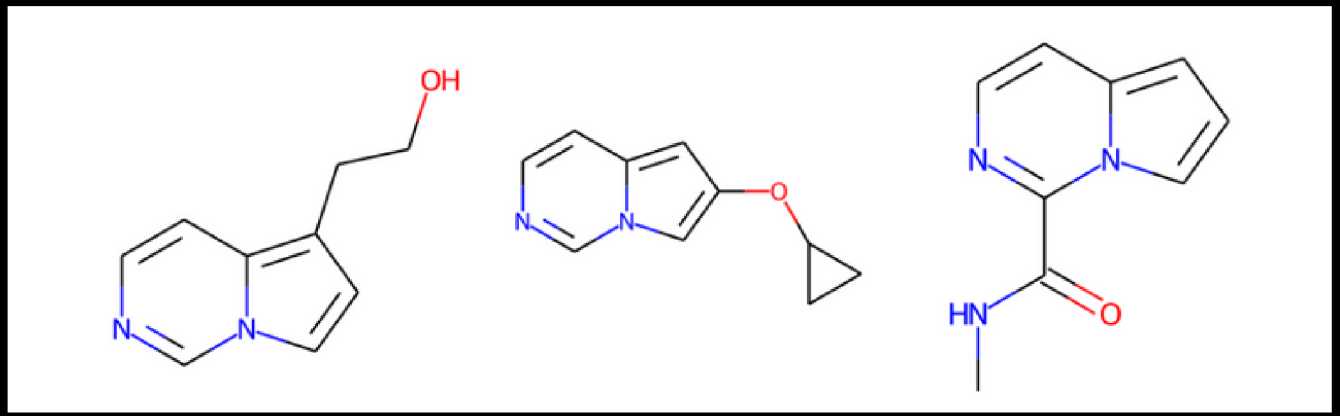
Biblioteka RDKit

C: > Users > Maja > Desktop > Nowy folder (2) > adssadas.py > ...

```python
from rdkit import Chem
from rdkit.Chem import Draw

m = Chem.MolFromSmiles('Cc1ccccc1')
#lub możemy również pobrać cząsteczkę/związek z pliku .mol
m = Chem.MolFromMolFile('data/input.mol')

img = Draw.MolToImage(m)
img.show()
```



SciByteHub



SciByteHub

```
def fib(n):
    result = []
    a, b = 0, 1
    while a < n:
        result.append(a)
        a, b = b, a+b
    return result
```

Definicja funkcji; n - ilość liczb w ciągu fibonacciego, które chcemy wyliczyć

Pusta lista, do której w pętli będziemy dodawać kolejne wartości ciągu fibonacciego

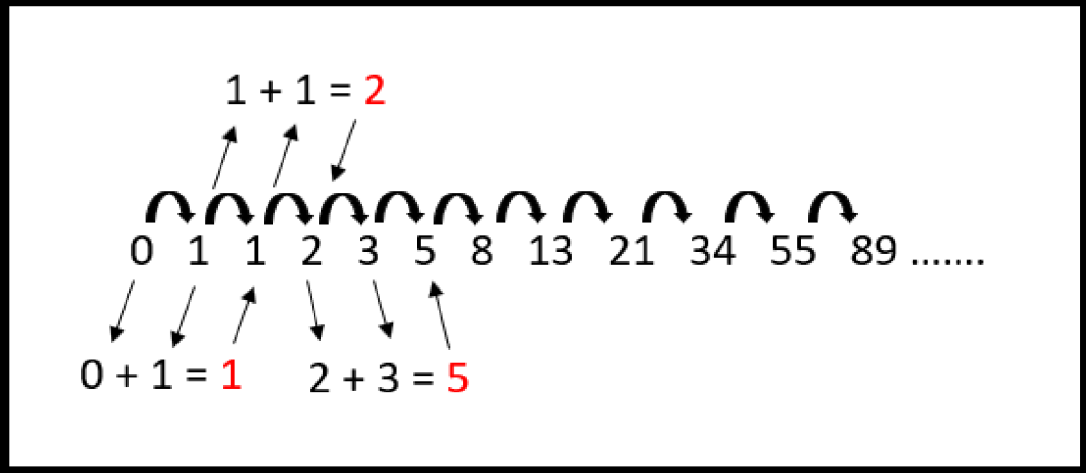
Deklaracja zmiennych a i b w celu wyliczania kolejnych wartości ciągu

Dopóki **a** jest mniejsze od **n** wykonuj pętlę

Dodanie do listy **result** a i b

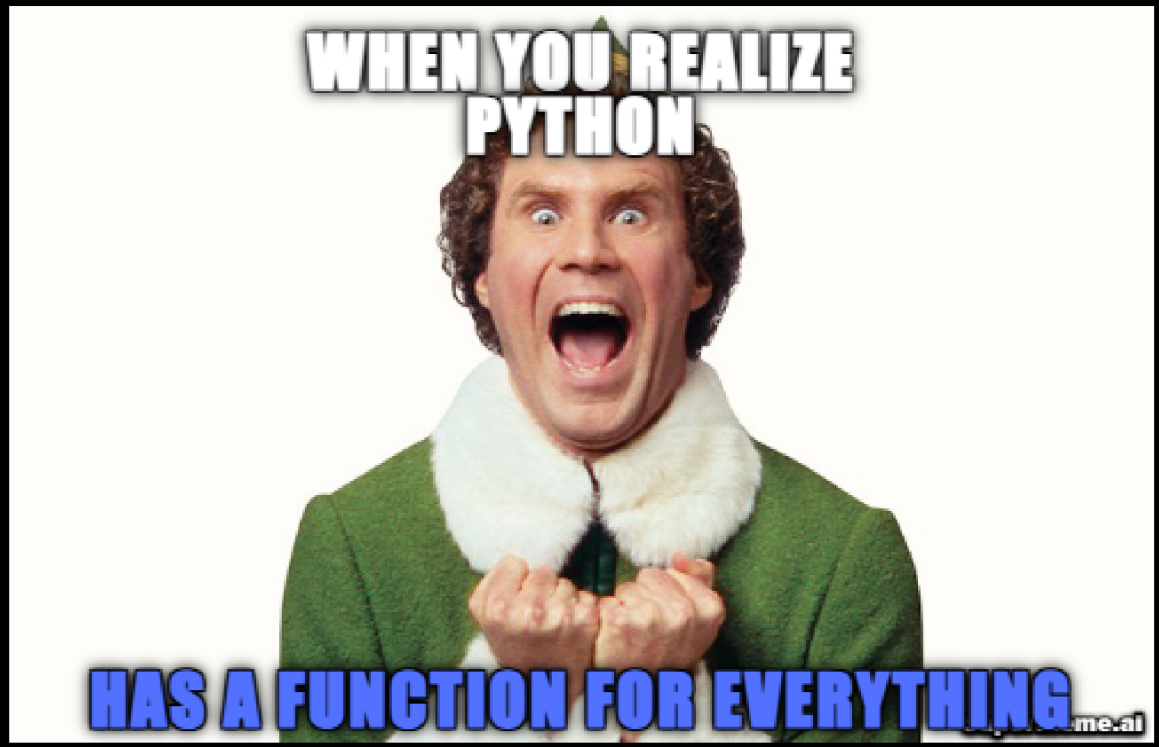
Wyliczenie kolejnych wartości a i b

Po ukończeniu pętli zwrócenie listy



SciByteHub

```
from sympy import fibonacci

def fib(n):
    result = []
    for i in range(n):
        result.append(fibonacci(i))
    return result
```



SciByteHub

C: > Users > Maja > Desktop > Nowy folder (2) > from sympy import fibonacci.py > ...

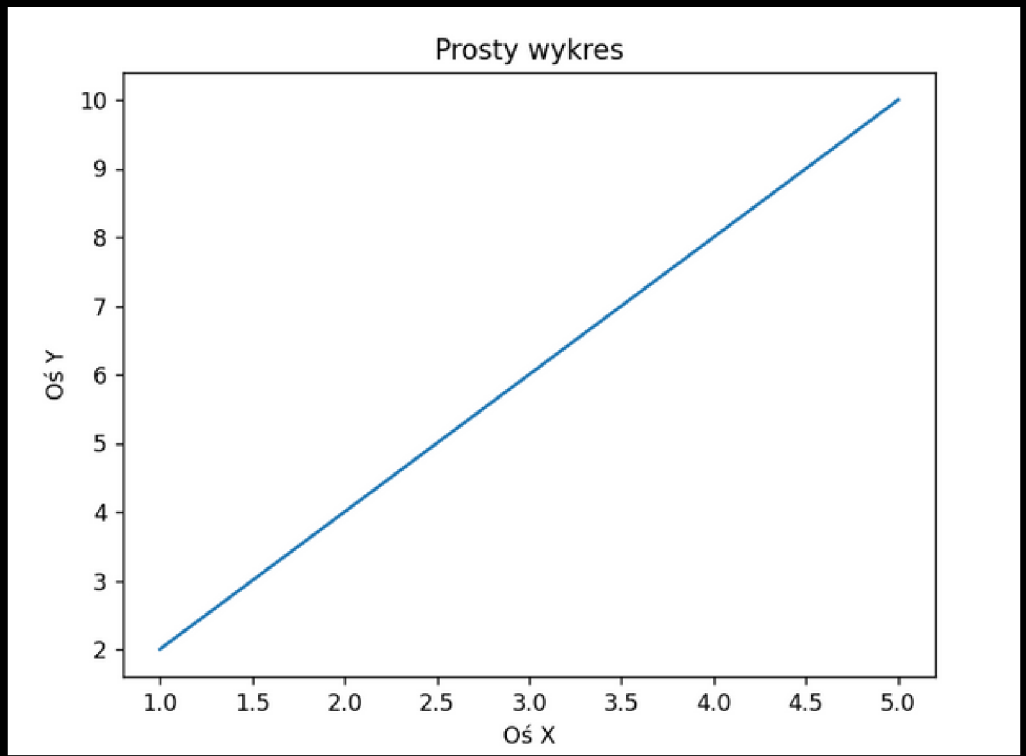
```python
from sympy import fibonacci

def fib_library(n):
    result = []
    for i in range(n):
        result.append(fibonacci(i))
    return result

print(fib_library(10))

def fib(n):
    result = []
    a, b = 0, 1
    while len(result) < n:
        result.append(a)
        a, b = b, a + b
    return result

print(fib(10))
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```


SciByteHub
Biblioteka matplotlib

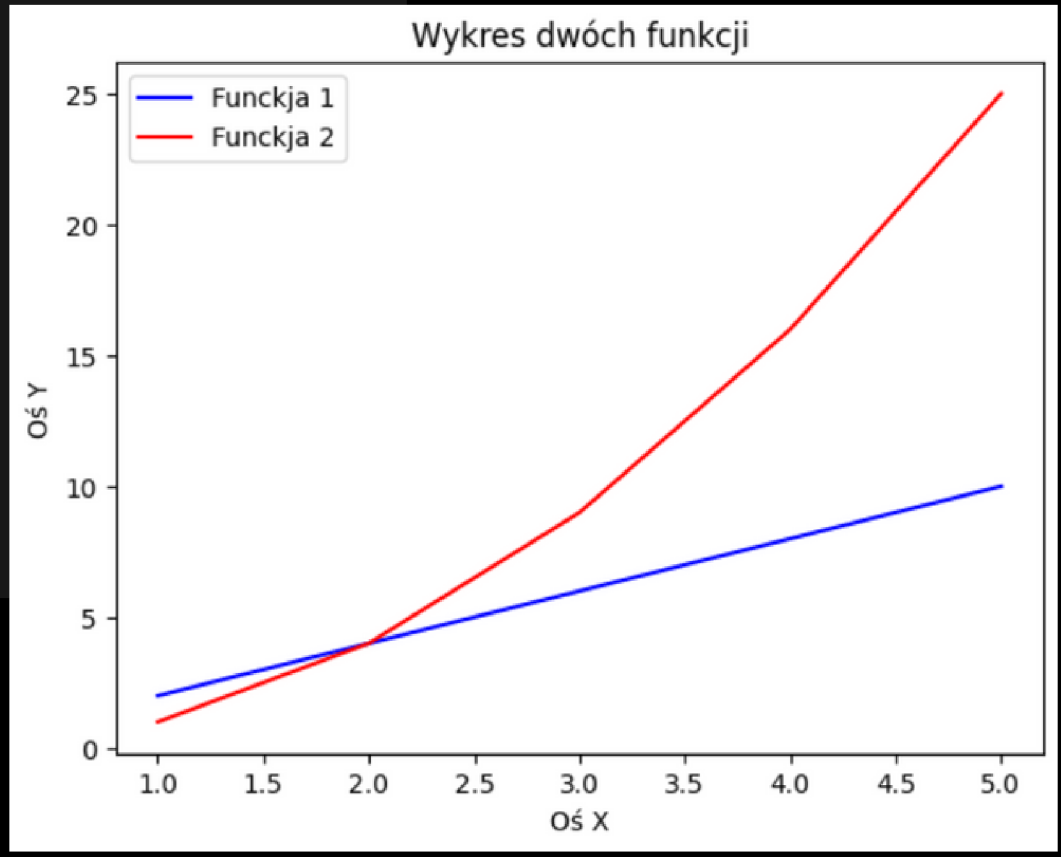
SciByteHub

```python
import matplotlib.pyplot as plt

# Przykładowe dane
x = [1, 2, 3, 4, 5]  # Przykładowa lista liczb dla osi x
y = [2, 4, 6, 8, 10]  # Przykładowa lista liczb dla osi y

# Tworzenie wykresu
plt.plot(x, y)

# Dodanie tytułu i etykiet osi
plt.title('Prosty wykres')
plt.xlabel('Oś X')
plt.ylabel('Oś Y')

# Wyświetlenie wykresu
plt.show()
```



SciByteHub

```python
import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y1 = [2,4,6,8,10]
y2 = [1,4,9,16,25]

# Pierwszy wykres
plt.plot(x,y1, label = "Funckja 1", color= 'blue')

# Dodanie drugiego wykresu
plt.plot(x,y2, label = "Funckja 2", color = 'red')

# Dodanie tytułu i opisów osi
plt.title('Wykres dwóch funkcji')
plt.xlabel('Oś X')
plt.ylabel('Oś Y')

# Dodanie legendy
plt.legend()

# Wyświetlenie wykresu
plt.show()
```



black
dimgray
dimgrey
gray
grey
darkgray
darkgrey
silver
lightgray
lightgrey
gainsboro
whitesmoke
white
snow
rosybrown
lightcoral
indianred
brown
firebrick
maroon
darkred
red
mistyrose
salmon
tomato
darksalmon
coral
orangered
lightsalmon
sienna
seashell
chocolate
saddlebrown
sandybrown
peachpuff
peru
linen

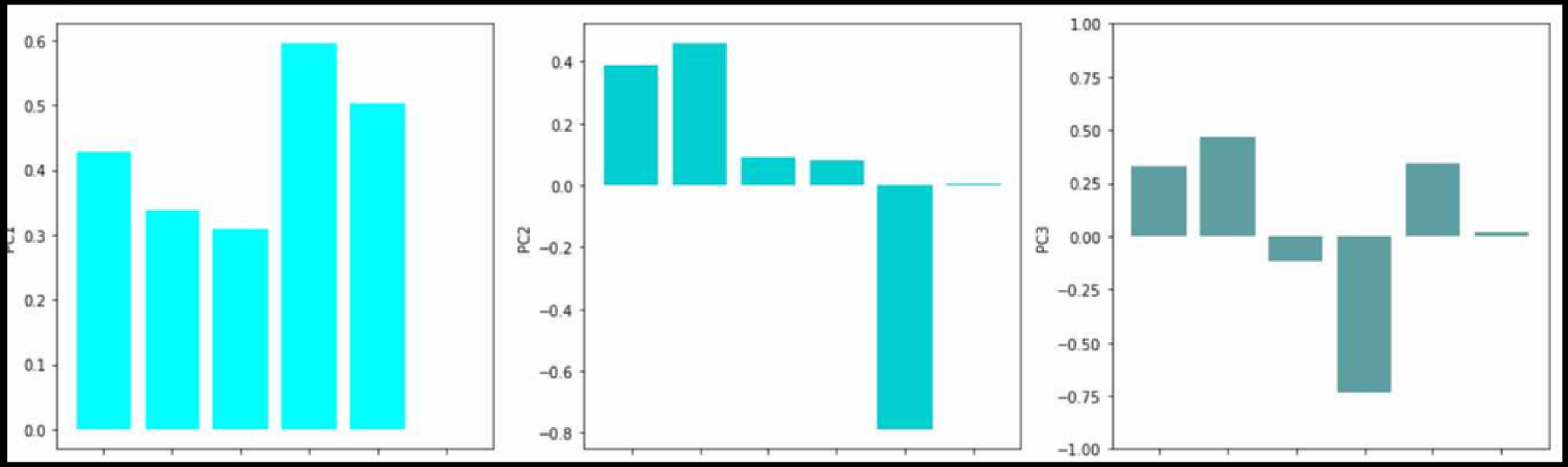
bisque
darkorange
burlywood
antiquewhite
tan
navajowhite
blanchedalmond
papayawhip
moccasin
orange
wheat
oldlace
floralwhite
darkgoldenrod
goldenrod
cornsilk
gold
lemonchiffon
khaki
palegoldenrod
darkkhaki
ivory
beige
lightyellow
lightgoldenrodyellow
olive
yellow
olivedrab
yellowgreen
darkolivegreen
greenyellow
chartreuse
lawngreen
honeydew
darkseagreen
palegreen
lightgreen

forestgreen
limegreen
darkgreen
green
lime
seagreen
mediumseagreen
springgreen
mintcream
mediumspringgreen
mediumaquamarine
aquamarine
turquoise
lightseagreen
mediumturquoise
azure
lightcyan
paleturquoise
darkslategray
darkslategrey
teal
darkcyan
aqua
cyan
darkturquoise
cadetblue
powderblue
lightblue
deepskyblue
skyblue
lightskyblue
steelblue
aliceblue
dodgerblue
lightslategray
lightslategrey
slategray

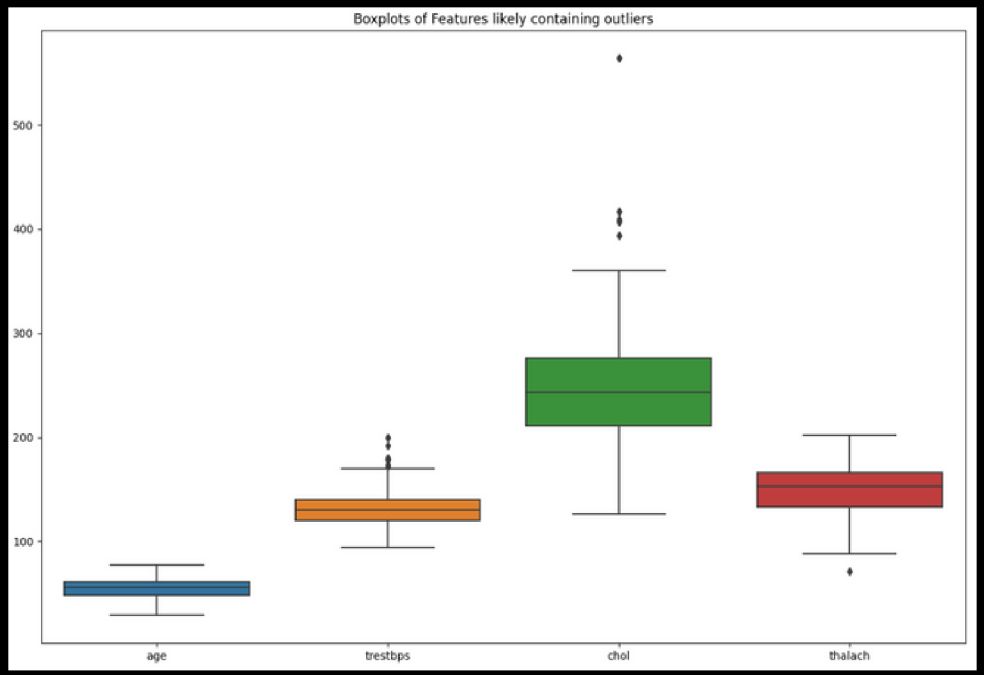
slategrey
lightsteelblue
cornflowerblue
royalblue
ghostwhite
lavender
midnightblue
navy
darkblue
mediumblue
blue
slateblue
darkslateblue
mediumslateblue
mediumpurple
rebeccapurple
blueviolet
indigo
darkorchid
darkviolet
mediumorchid
thistle
plum
violet
purple
darkmagenta
fuchsia
magenta
orchid
mediumvioletred
deeppink
hotpink
lavenderblush
palevioletred
crimson
pink
lightpink

SciByteHub

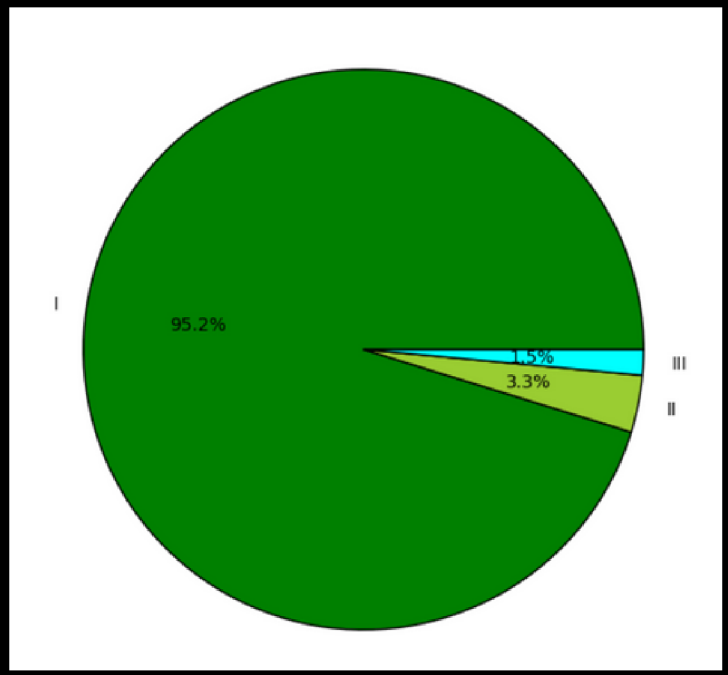
bar plot (subplot)



boxplot

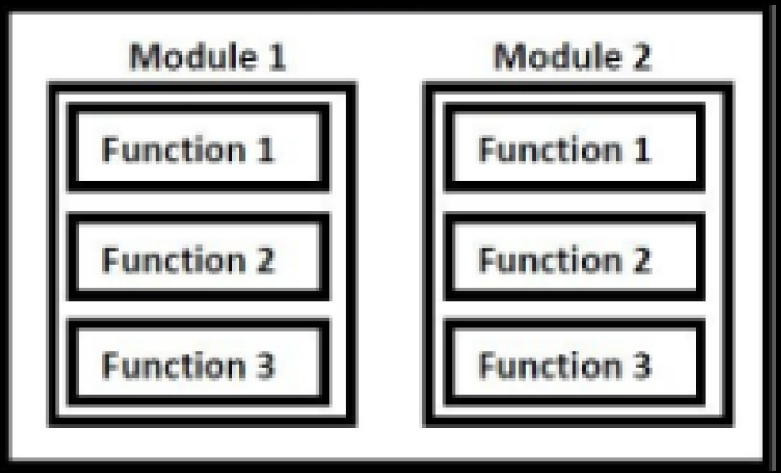


pie chart

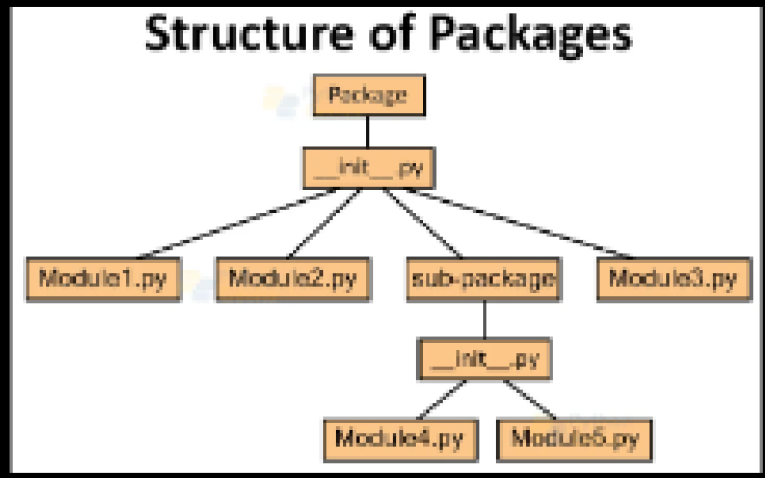


SciByteHub

Moduły



Biblioteki



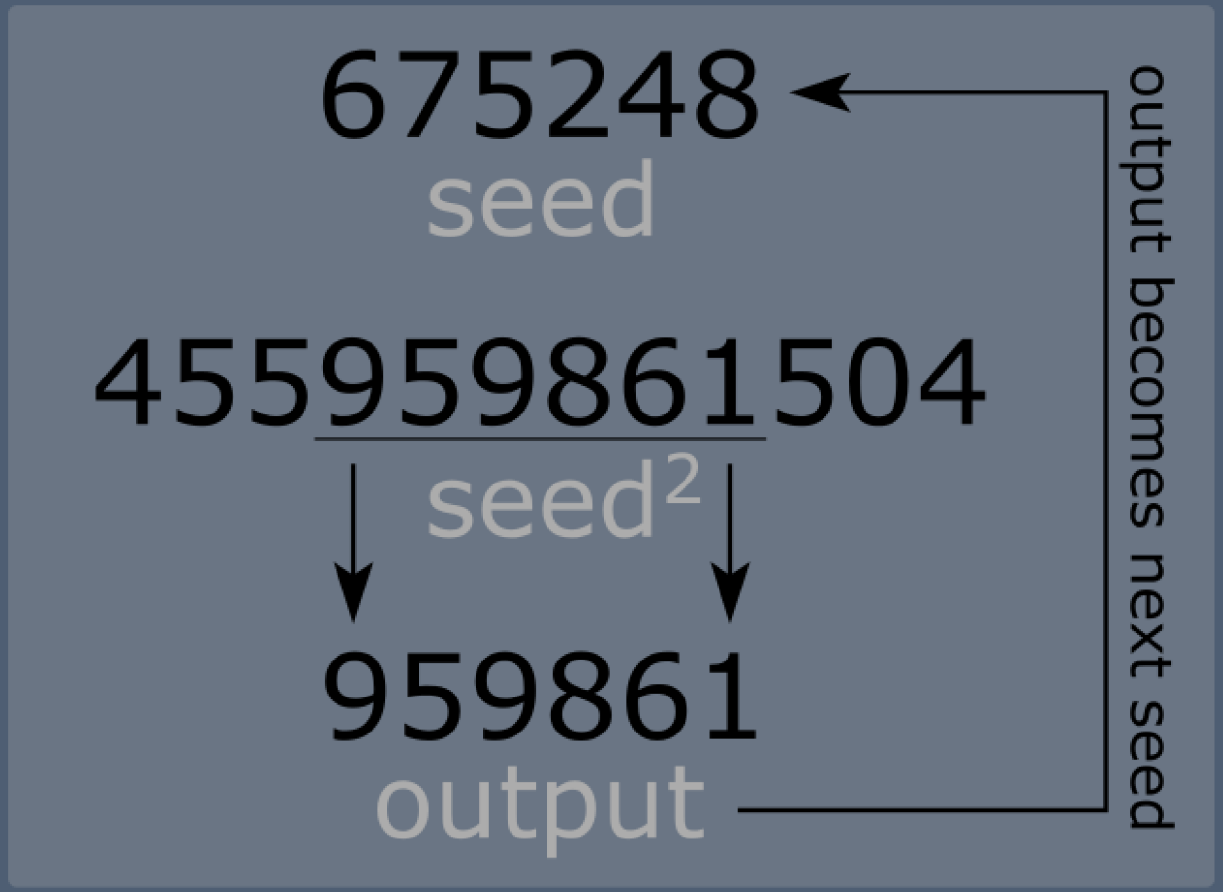
SciByteHub

Generatory liczb pseudolosowych

Przykłady generatorów :

randint(0,10)	Zwraca liczbę naturalną z zadanego zakresu (np. od 0 do 10)
random()	Zwraca liczbę rzeczywistą między 0 a 1
uniform(0,10)	Zwraca liczbę rzeczywistą z zadanego zakresu (np. od 0 do 10)

Generatory liczb pseudolosowych



Jeśli nie ustawimy ziarna -> zostanie ono ustawione na podstawie czasu systemowego


```
>>> prng1(9)
[0, 7, 2, 9, 4, 11, 6, 1, 8]
```

```
def prng1(n):
  seq = [0]          # seed (starting value)
  for i in range(1, n):
     seq.append((seq[-1] + 7) % 12)
  return seq
```

Użycie modułu random i funkcji randint

```python
import random

r = random.randint(1,10)
print('Losowa liczba między 1 a 10:', r)
```

```
Losowa liczba między 1 a 10: 3
```

```
Losowa liczba między 1 a 10: 7
```

```
Losowa liczba między 1 a 10: 2
```

```python
import random
import numpy as np

random.seed(1000)
r = random.randint(1,10)
print('Losowa liczba między 1 a 10:', r)
```

```
Losowa liczba między 1 a 10: 7
```

```
Losowa liczba między 1 a 10: 7
```

SciByteHub

Wykorzystanie pętli do stworzenia listy 10, losowych liczb z zakresu od 1 do 100

```python
import random

lista=[]
for i in range(10):
    i = random.randint(1,100)
    lista.append(i)

print(lista)
```

```
[47, 30, 59, 24, 6, 87, 96, 63, 19, 60]
```

SciByteHub

Przykłady z życia

- Gry komputerowe np. minecraft

Po prowadzeniu wartości seeda gra przekazuje seed do generatora liczb pseudolosowych (PRNG) i używa do generowania świata.

Podanie tego samego nasiona da tę samą sekwencję z PRNG, a tym samym wygeneruje ten sam świat.

- Internetowe gry hazardowe
- Rynek finansowy

SciByteHub

Macierze i ich mnożenie

$$3 \cdot 0 + 1 \cdot 2 + 0 \cdot 0 = 2$$

$$\begin{bmatrix} 3 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 4 \\ 2 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & \\ & \end{bmatrix}$$

2×3 $\quad$ 3×2 $\quad$ 2×2

SciByteHub

Macierze w Pythonie

```python
import numpy as np

A = np.array([[1, 2, 3], [4, 5, 6]])
print(A)
```

✓ 0.0s

```
[[1 2 3]
 [4 5 6]]
```

```python
import numpy as np


zerowa_macierz = np.zeros((2,3))
print(zerowa_macierz)
```

✓ 0.0s

```
[[0. 0. 0.]
 [0. 0. 0.]]
```

```python
import numpy as np

print('B:')
B = np.arange(4)
print(B)

print("C:")
C = np.arange(6).reshape(2, 3)
print(C)
```

✓ 0.0s

```
B:
[0 1 2 3]
C:
[[0 1 2]
 [3 4 5]]
```

SciByteHub

Mnożenie macierzy

```
import numpy as np

A = np.array([[3,1,0],[0,2,1]])
B = np.array([[0,4],[2,2],[0,1]])

np.dot(A,B)
```

✓ 0.0s

```
array([[ 2, 14],
       [ 4,  5]])
```

SciByteHub

Transponowanie macierzy

$$A = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}_{2 \times 3} \qquad A^T = \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}_{3 \times 2}$$

SciByteHub

Transponowanie macierzy w Pythonie

```python
import numpy as np

print("Macierz A:")
A = np.array([[1, 2, 3], [4, 5, 6]])
print(A)

print("Transponowana macerz A")
A_transponowane = A.transpose()
print(A_transponowane)
```

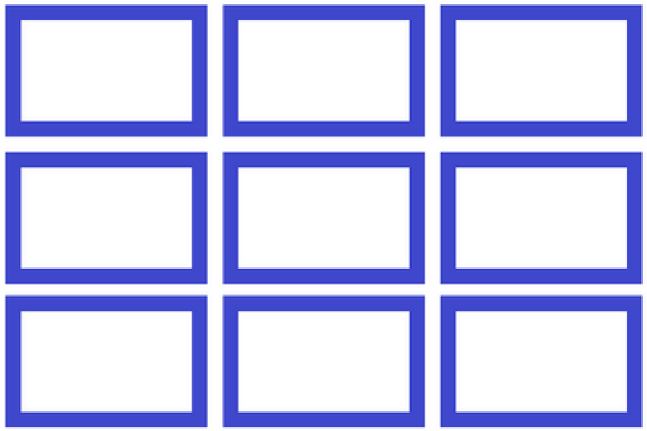
✓ 0.0s

```
Macierz A:
[[1 2 3]
 [4 5 6]]
Transponowana macerz A
[[1 4]
 [2 5]
 [3 6]]
```

SciByteHub

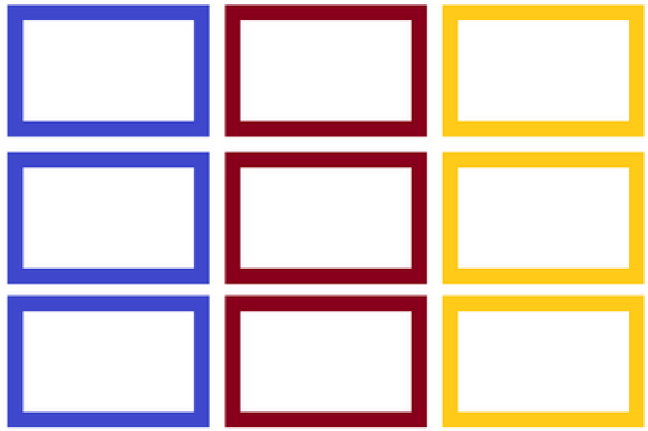
Macierz kontra Data Frame

Macierz



Przechowuje tylko dane numeryczne

Data Frame



Przechowuje zarówno dane numeryczne, jak i stringi lub dane logiczne (boolean type - True, False)

SciByteHub

Using Python for Numpy and implicit data types

```
TypeError: only integer scalar arrays can be converted to a scalar index
```

```
ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or a.all()
```

SciByteHub