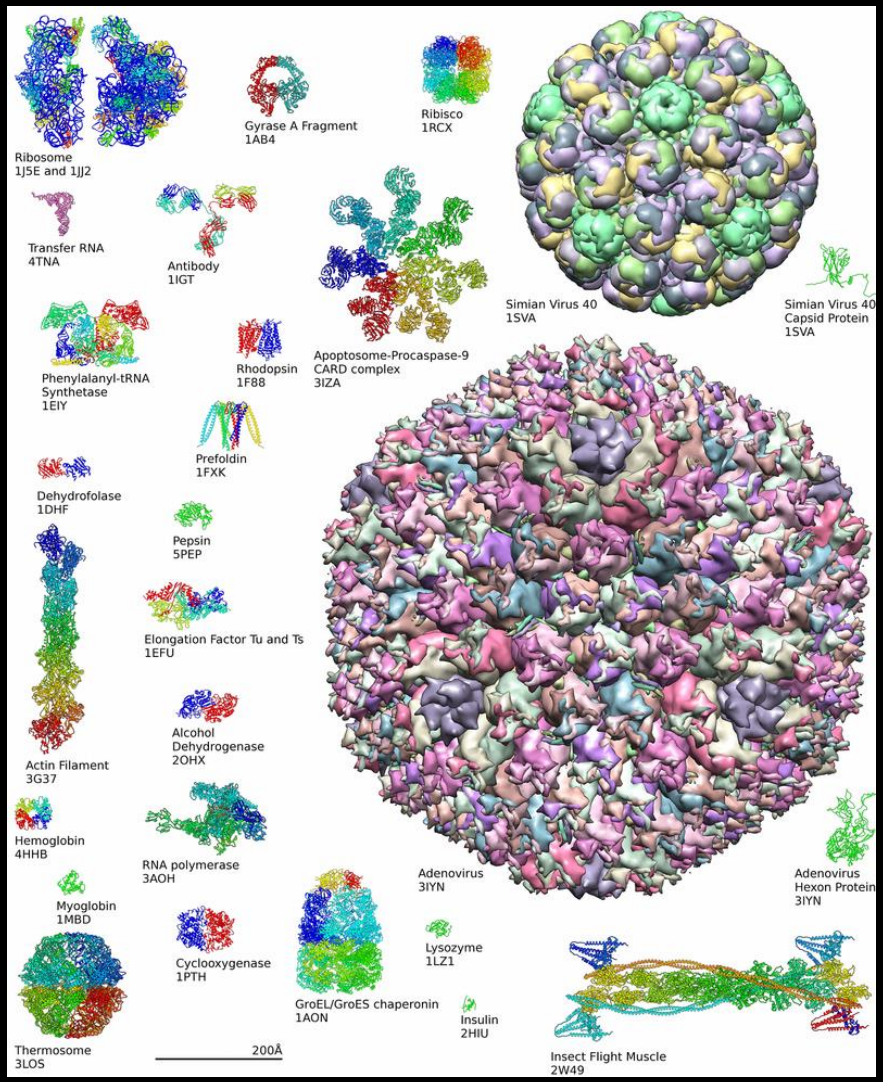
# Analiza białek z wykorzystaniem plików PDB

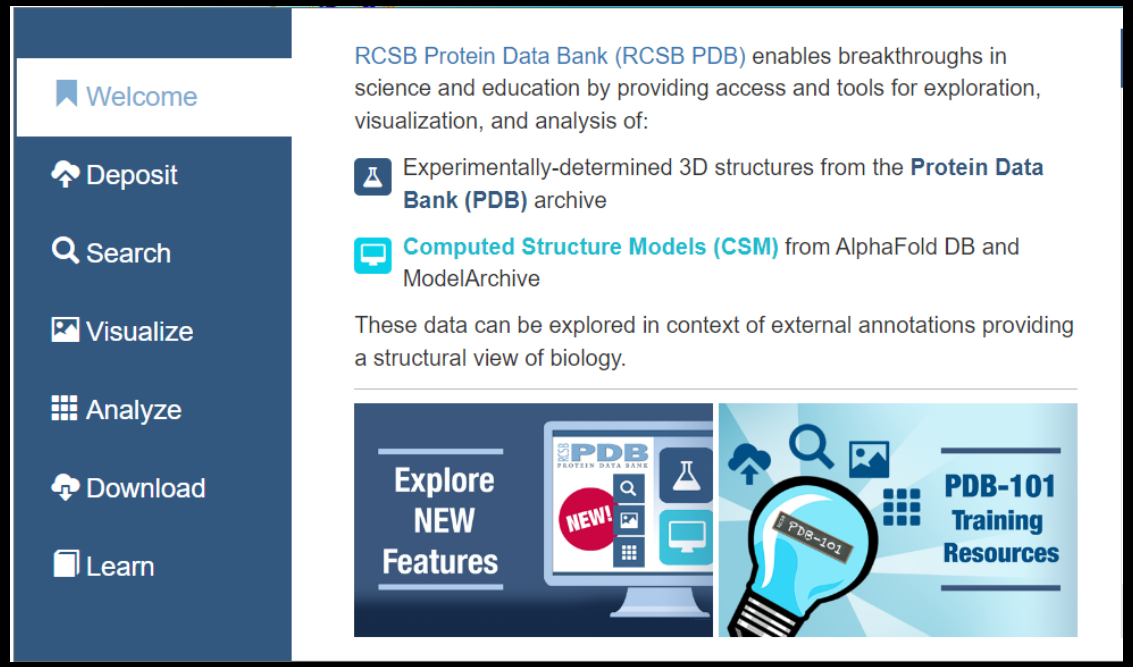
Skład zespołu:
Adrian Szczyrba
Adam Borys
Adrianna Cyraniak

Koordynator:
Marta Pągielska

11.06.2024

SciByteHub




# Baza PDB

# Protein Data Bank [edytuj]



**Protein Data Bank**, w skrócie **PDB** jest to biologiczna baza danych gromadząca dane o strukturze przestrzennej białek i kwasów nukleinowych.

Dane o tych makrocząsteczkach uzyskane m.in. za pomocą rentgenografii strukturalnej i spektroskopii NMR zawierają bibliografię, sekwencje aminokwasowe, informacje o strukturze drugorzędowej oraz, przede wszystkim, współrzędne atomów.

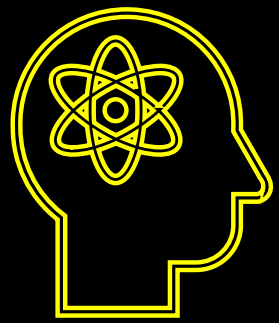
Na dzień 26 października 2020 r. baza PDB zawierała 170 171 następujących struktur[1]:

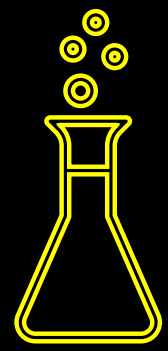
| Metoda | Białka | Kwasy nukleinowe | Kompleksy białko/kwas nukleinowy | Inne | W sumie |
|---|---|---|---|---|---|
| Rentgenografia strukturalna | 141 144 | 2 158 | 7 188 | 4 | 150 494 |
| Spektroskopia NMR | 11 519 | 1 362 | 265 | 8 | 13 154 |
| Mikroskopia elektronowa | 4 491 | 47 | 1 478 | 0 | 6 016 |
| Metody mieszane | 167 | 6 | 3 | 1 | 177 |
| Inne | 307 | 4 | 6 | 13 | 330 |
| **W sumie** | 157 628 | 3 3577 | 8 940 | 27 | **170 171** |

SciByteHub

# Cel

Celem naszego projektu było zautomatyzowanie analizy białek. Dzięki napisanym przez nas funkcjom użytkownik może sprawdzić kilka istotnych właściwości białka podając jedynie jego kod z bazy PDB. Zredukuje to czas, który użytkownik musiałby poświęcić na research .



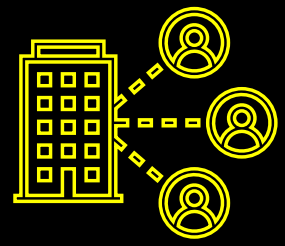


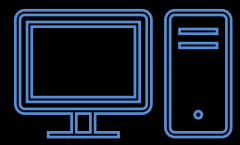
SciByteHub

# Co chcieliśmy uzyskać?

Zależało nam na przejrzeniu podstawowych właściwości białek i przedstawieniu ich w atrakcyjny sposób.
Stąd konieczne było: zainstalowanie i zaimportowanie bibliotek i narzędzi służących do:

- wizualizacji,
- wykresów
- obliczeń matematycznych.





Każdy z nas zaproponował kilka funkcjonalności, które chcieliśmy sprawdzić i oddzielnie napisaliśmy swoje fragmentu kodu.

SciByteHub

# Metody i narzędzia

```
# biblioteki
from rdkit import Chem
from rdkit.Chem import Draw
from termcolor import colored
from Bio import pairwise2
from Bio.pairwise2 import format_alignment
import os
import numpy as np
import matplotlib.pyplot as plt
import requests
from Bio import SeqIO
from Bio.SeqUtils.ProtParam import ProteinAnalysis
from Bio.PDB.PDBParser import PDBParser
from ramachandraw.utils import fetch_pdb, plot
import json
import joblib
from Bio import PDB
import icn3dpy
from IPython.display import display
import py3Dmol
```

GitHub

Google colab

Visual Studio Code

jupyter

Korzystaliśmy z konstrukcji funkcyjnych

SciByteHub

# Nasza współpraca

Adrian

Dobra dziubki

Ja już trochę popisałem

Wybrałem 3 funkcje nie licząc api

1. Zapis sekwencji do pliku

2. Wykres procentowy aminokwasów w sekwencji

3. Wykres Ramachandrana

Przesyłam link do githuba

dotarłam do momentu że printuje xd

Ale nie liczy

Adrian

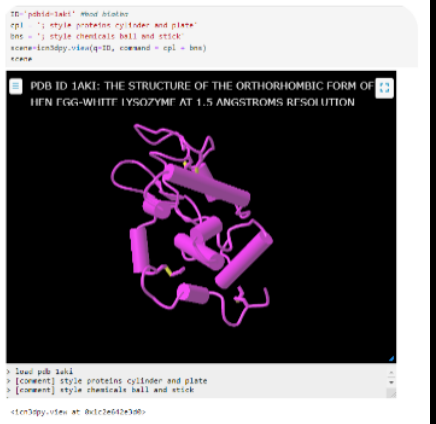
Hej, dodałem update na git

sprawdzcie czy jest?

Mogę coś pisać w kodzie czy planujecie zmiany jakieś?

Później bym tylko pushnął

zamknęłam w funkcję i update'uję




dobra, działa, jest dość krótkie, ale chyba się nada co?

Adam

Dobra to ja wymyśliłem swoje i to by były: Narysowanie z użyciem rdkit aminokwasu, którego jest najwięcej i wyświetlenie jego właściwości; porównanie sekwencji z innym białkiem (procent homologii); określenie i uwidocznienie obszarów hydrofobowych/ hydrofilowych w białku

O to hydrofobowość i hydrofobowość super

Hydrofilowosc*

Adam

No tylko mysle jakby to ugryźć haha

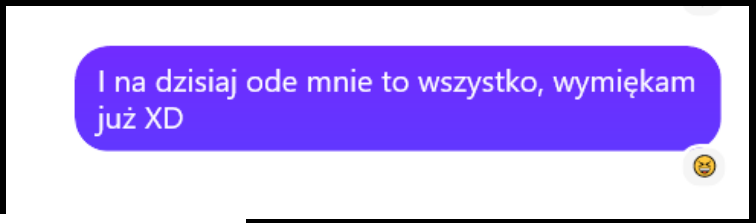
Adam

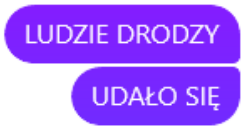
Ej nice

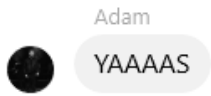
Fajowe 😈

SciByteHub

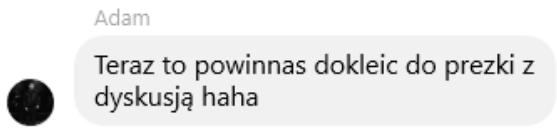
# Nasza współpraca

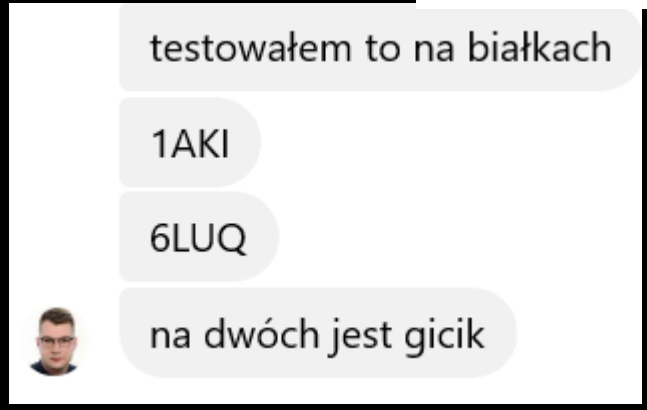



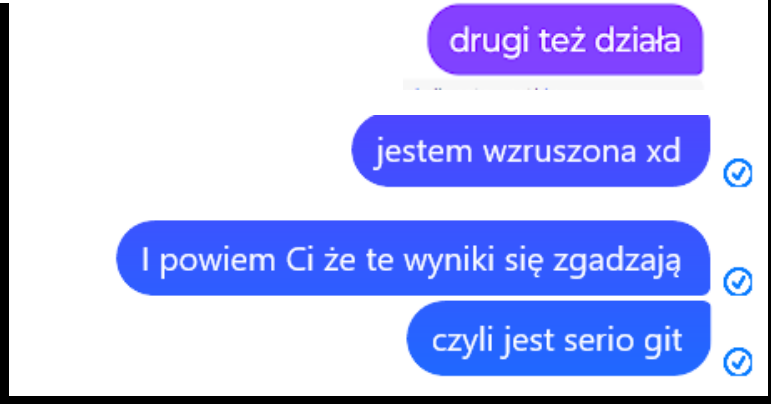












# Kod

```
#os things
pdb_entry = input("Wprowadź PDB ID białka które chcesz zbadać: ")
os.mkdir(pdb_entry) #tworzy folder o nazwie pdb_entry
os.chdir(pdb_entry) #zmienia PWD na folder pdb_entry, wszystkie dane będą zapisywane w tym folderze

# pierwsza funkcja, pobiera PDB
def Download(pdb_entry):
    url = f"https://files.rcsb.org/download/{pdb_entry}.pdb"
    response = requests.get(url)
    if response.status_code == 200:
        with open(f"{pdb_entry}.pdb", "wb") as file:
            file.write(response.content)
            print(f"Plik został pobrany pomyślnie w folderze {pdb_entry}!")
    else:
        print("Nie udało się pobrac pliku!")
Download(pdb_entry)
```

```
# druga funkcja zapisuje sekwencję białka do pliku seq_pdb_entry.txt
def GetSequence(pdb_entry):
    sequence = f'{pdb_entry}.pdb'
    for sequence in SeqIO.parse(sequence, "pdb-seqres"):
        file1 = open(f'seq_{pdb_entry}.txt', 'w')
        file1.write(f'Sekwencja białka {pdb_entry}:\n {sequence.seq}')
        file1.close()
        print(f'Sekwencja została pomyślnie zapisana do pliku seq_{pdb_entry}.txt!')
GetSequence(pdb_entry)
```


# Kod

```python
#trzecia funkcja liczy częstotliwość pojawiania się aminokwasu w sekwencji i zapisuje wykres do pliku Plot1.png
def AAcount(pdb_entry):
    sequence = f'{pdb_entry}.pdb'
    for sequence in SeqIO.parse(sequence, "pdb-seqres"):
        sequence = str(sequence.seq)
        analysed_sequence = ProteinAnalysis(sequence)
        aacount = analysed_sequence.count_amino_acids()
        labels = list(aacount.keys())
        sizes = list(aacount.values())
        max_amino_acid = max(aacount, key=aacount.get)
        fig, ax = plt.subplots(figsize=(10,10))
        ax.pie(sizes,labels=labels,autopct='%1.1f%%', pctdistance=1.2, labeldistance=0.6, startangle=90)
        plt.title(f'Udział procentowy aminokwasów w sekwencji białka {pdb_entry}')
        plt.savefig(f'Sequence_plot_{pdb_entry}.png')
        print(f'Procentowa zawartość aminokwasów została pomyślnie zapisana na wykresie pliku Sequence_plot_{pdb
    aa_molecule = Chem.MolFromSmiles(aminokwasy[max_amino_acid])
    image = Draw.MolToImage(aa_molecule, size=(300, 300))
    image.show()

AAcount(pdb_entry)
```

```python
#czwarta funkcja rysuje wykres Ramachandrana na podstawie danych w PDB
def RamachandranPlot(pdb_entry):
    ram_plot = f"{pdb_entry}.pdb"
    plot(ram_plot, save=True, show=False, filename=f'Ramachandran_plot_{pdb_entry}.png')
    print(f'Wyres Ramachandrana został pomyślnie zapisany do pliku Ramachandran_plot_{pdb_entry}.png!')

RamachandranPlot(pdb_entry)
```


# Kod

```
# określenie progu hydrofobowości
hydrophobic_aa = []
hydrophylic_aa = []


# pokolorowanie symboli aminokwasów w zależności od ich hydrofobowości/hydrofilowości

def FragmentyHydrofobowe(pdb_entry):
    threshold = float(input('Podaj granicę hydrofobowości: '))
    for aa, hydrophobicity_value in hydrophobicity_scale.items():
      if hydrophobicity_value > threshold:
        hydrophobic_aa.append(aa)
      else:
        hydrophylic_aa.append(aa)

    phobic_phylic_peptide_sequence = ''
```

```
    sequence = f'{pdb_entry}.pdb'
    for sequence in SeqIO.parse(sequence, "pdb-seqres"):
        sequence = str(sequence.seq)

    for i in sequence:
      if i in hydrophobic_aa:
        phobic_phylic_peptide_sequence += (colored(i, 'red'))
      else:
        phobic_phylic_peptide_sequence += (colored(i, 'blue'))
    print(phobic_phylic_peptide_sequence)

FragmentyHydrofobowe(pdb_entry)
```


# Kod

```python
#Obliczanie ilości reszt aminokwasowych
def ObliczLiczbaResztAminokwasowych(pdb_entry):
  parser=PDB.PDBParser()
  protein=parser.get_structure('bialko', f'{pdb_entry}.pdb')

#liczanie reszt aminowkasowych
  residues=set()
  for model in protein:
    for chain in model:
      for residue in chain:
        if PDB.is_aa(residue, standard=True):
          residues.add(residue.id)

  print(f'W białku jest {len(residues)} reszt aminokwasowych')

ObliczLiczbaResztAminokwasowych(pdb_entry)
```

```python
#Obliczanie dlugosci bialka w angstremach
def ObliczDlugoscBialka(pdb_entry):
  parser=PDB.PDBParser()
  protein=parser.get_structure('bialko', f'{pdb_entry}.pdb')

#wyznaczenie "krawędzi" białka
  min_coord = np.array([np.inf, np.inf, np.inf])
  max_coord = np.array([-np.inf, -np.inf, -np.inf])

#przeliczanie po strukturze
  for model in protein:
    for chain in model:
      for residue in chain:
        for atom in residue:
          coord=atom.coord
          min_coord = np.minimum(min_coord, coord)
          max_coord = np.maximum(max_coord, coord)

#wyznaczenie wektora odleglosci i jego zmierzenie
  dlugosc = max_coord - min_coord
  DlugoscBialka=np.linalg.norm(dlugosc)

  print(f"Dlugosc bialka: {DlugoscBialka:2f} Å")

ObliczDlugoscBialka(pdb_entry)
```

SciByteHub

# Kod

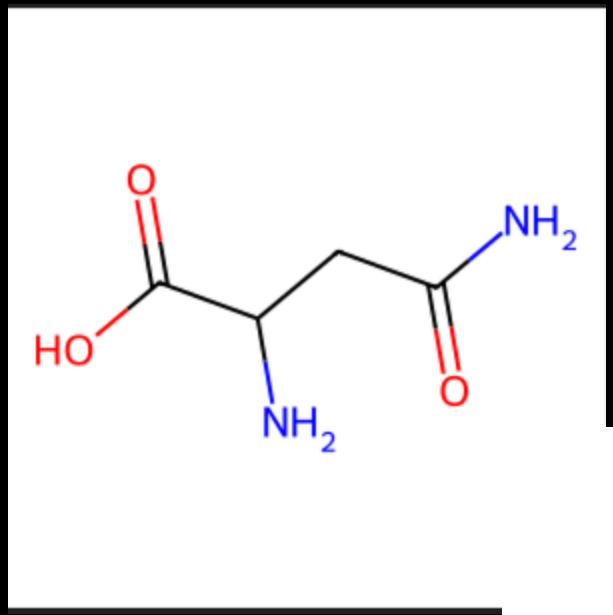
```
#wizualizacja białka

def WizualizacjaStrukturyBialka(pdb_entry, protein_style='cylinder and plate', ligand_style='ball and stick'):
    style = f'bialko {protein_style}; ligand {ligand_style}'

    # Tworzenie widoku 3D
    scene = icn3dpy.view(q=pdb_entry, command=style)

    return scene

# Otworzenie wizualizacji
scene=WizualizacjaStrukturyBialka(f"pdbid={pdb_entry}")
display(scene)
```

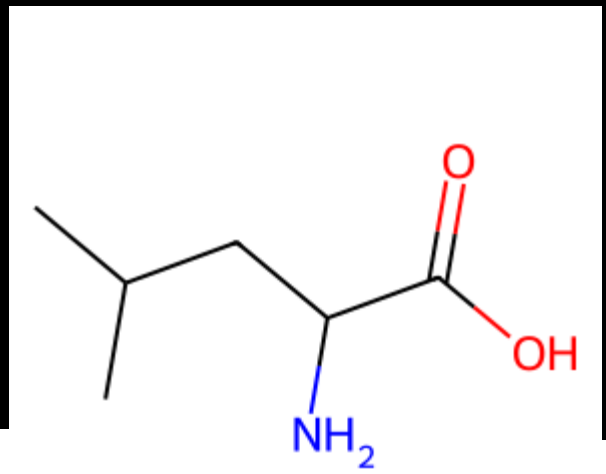
SciByteHub

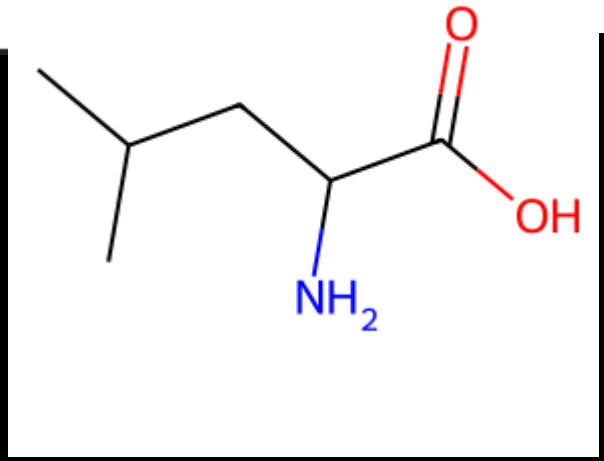
# Demo

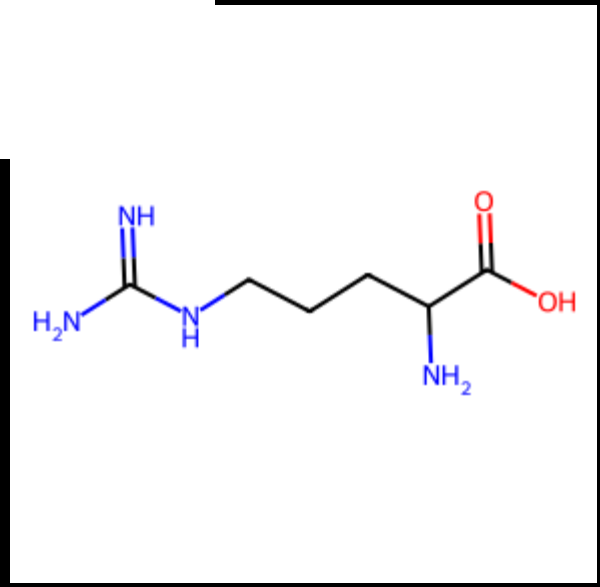
Działanie kodu:

https://drive.google.com/file/d/1CcYzfJqSm_y6Hcf3FeZVykAQXYM95d2H/view?usp=sharing



SciByteHub

# Wnioski

- Większość założonych funkcji udało się uzyskać.
- Możliwości jest znacznie więcej, baza dysponuje wieloma danymi.
- Wizualizacja działa w jupyterlab, w innych środowiskach kod działa, ale nie wyświetla się obraz.
- Można było zapewnić lepszą obsługę błędów, ale ograniczył nas czas.

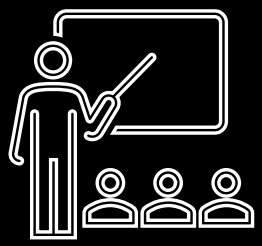
SciByteHub

# DZIĘKUJEMY ZA UWAGĘ!

Kołu SciByteHub dziękujemy za zorganizowanie kursu,

a Marcie Pągielskiej dziękujemy za pomoc w przygotowaniu projektu ☺

Ada, Adam i Adrian



SciByteHub

SciByteHub