

```
1
2
3  Praktyczne_zastosowanie_Pythona = {
4
5      w_naukach : ['biologicznych', 'i medycznych'],
6      dla :      ['początkujących'],
7      poziom :   ['drugi'],
8      część :    ['trzecia'],
9
10
11      # Mateusz Dobrychłop, 28 listopada 2023
12  }
```

```
1 harmonogram_listopad = {
```

```
2     M    T    W    T    F    S    S
```

```
3         01  02  03  04  05
```

```
4         06  07  08  09  10  11  12
```

```
5         13  14  15  16  17  18  19      Spotkanie 1 (17:00 – 19:30)
```

```
6         20  21  22  23  24  25  26      Spotkanie 2 (17:00 – 19:30)
```

```
7         27  28  29  30      Spotkanie 3 (17:00 – 19:30)
```

```
8     }
```

```
1 Spotkanie_02 = {
```

```
2  
3     01 pandas i matplotlib - rozwinięcie
```

```
4  
5  
6  
7     02 OpenAI API
```

```
8  
9  
10  
11  
12  
13  
14 }
```



1. Otwieramy linię komend (cmd, PowerShell, terminal itp.)
2. Przechodzimy do folderu z naszymi skryptami.
3. Aktywujemy środowisko wirtualne:



```
.\venv\Scripts\activate.ps1
```



```
source venv/bin/activate
```

```
pip install openai
```

```
PS C:\Users\Dobry\Documents\szkolenie2023> .\venv\Scripts\activate.ps1  
(venv) PS C:\Users\Dobry\Documents\szkolenie2023>
```



```
1  import sys
2
3  print(sys.argv)
4
```

```
PS C:\Szkolenie_2023\KOD> python sys_argv_01.py argument1 argument2 argument3
['sys_argv_01.py', 'argument1', 'argument2', 'argument3']
```

```
1  import sys
2
3  print(sys.argv)
4
```

```
PS C:\Szkolenie_2023\KOD> python sys_argv_01.py argument1 argument2 argument3
['sys_argv_01.py', 'argument1', 'argument2', 'argument3']
```

```
1 > import sys
2   import pandas as pd
3   import matplotlib.pyplot as plt
4
5 > def profile(dataframe, list_of_numeric_cols): ...
22
23 > def clean_missing_values(dataframe): ...
36
37 > def remove_duplicates(dataframe): ...
51
52 > def profiler_cleaner(file_path, *args): ...
78
79 # Przykład użycia: python data_profiler.py data.xlsx profile clean_missing_values
80 profiler_cleaner(sys.argv[1], *sys.argv[2:])
81
```

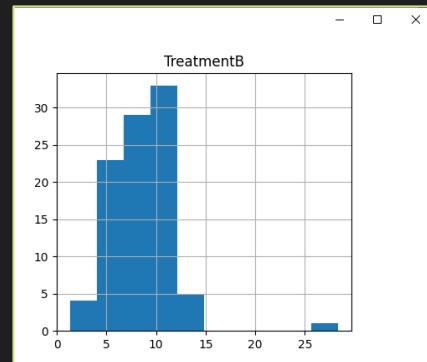
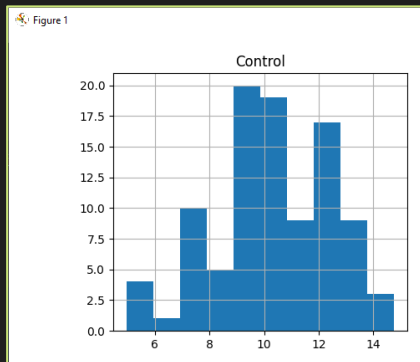
```
1 > import sys
2   import pandas as pd
3   import matplotlib.pyplot as plt
4
5 > def profile(dataframe, list_of_numeric_cols): ...
22
23 > def clean_missing_values(dataframe): ...
36
37 > def remove_duplicates(dataframe): ...
51
52 > def profiler_cleaner(file_path, *args): ...
78
79   # Przykład użycia: python data_profiler.py data.xlsx profile clean_missing_values
80   profiler_cleaner(sys.argv[1], *sys.argv[2:])
81
```



```
52 ~ def profiler_cleaner(file_path, *args):
53     # Wczytywanie danych
54     data = pd.read_excel(file_path)
55
56     # Ustalanie kolumn numerycznych
57     list_of_numeric_cols = []
58     for col in data.columns:
59         if data[col].dtype in ['int64', 'float64']:
60             list_of_numeric_cols.append(col)
61
62     # Przejście przez argumenty i wykonanie odpowiednich operacji
63     for arg in args:
64         if arg == 'profile':
65             profile(data, list_of_numeric_cols)
66
67         elif arg == 'clean_missing_values':
68             data = clean_missing_values(data)
69
70         elif arg == 'remove_duplicates':
71             data = remove_duplicates(data)
72
73
74     # Zapisanie oczyszczonych danych
75     clean_file_path = 'clean_' + file_path
76     data.to_excel(clean_file_path, index=False)
77     print(f"Oczyszczone dane zapisane w: {clean_file_path}")
78
79     # Przykład użycia: python data_profiler.py data.xlsx profile clean_missing_values
80     profiler_cleaner(sys.argv[1], *sys.argv[2:])
81
```

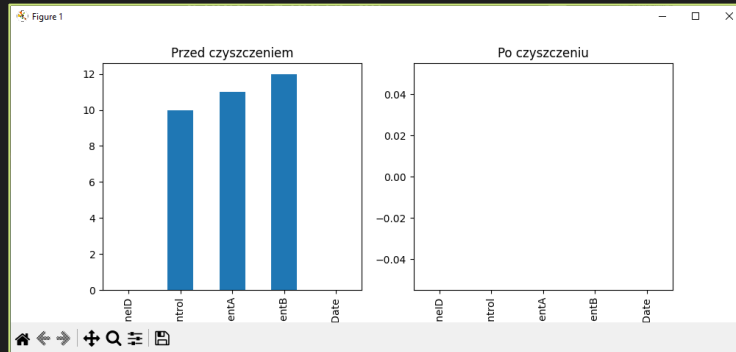
```
1 import sys
2 import pandas as pd
3 import matplotlib.pyplot as plt
4
5 > def profile(dataframe, list_of_numeric_cols): ...
22
23 > def clean_missing_values(dataframe): ...
36
37 > def remove_duplicates(dataframe): ...
51
52 > def profiler_cleaner(file_path, *args): ...
78
79 # Przykład użycia: python data_profiler.py data.xlsx profile clean_missing_values
80 profiler_cleaner(sys.argv[1], *sys.argv[2:])
81
```

```
5 def profile(dataframe, list_of_numeric_cols):
6     print("HEAD AND TAIL:")
7     print(dataframe.head())
8     print(dataframe.tail())
9     print("INFO:")
10    print(dataframe.info())
11    print("DESCRIBE:")
12    print(dataframe.describe())
13    # Wstępna wizualizacja
14    for num_col in list_of_numeric_cols:
15        dataframe[num_col].hist()
16        plt.show()
17
```



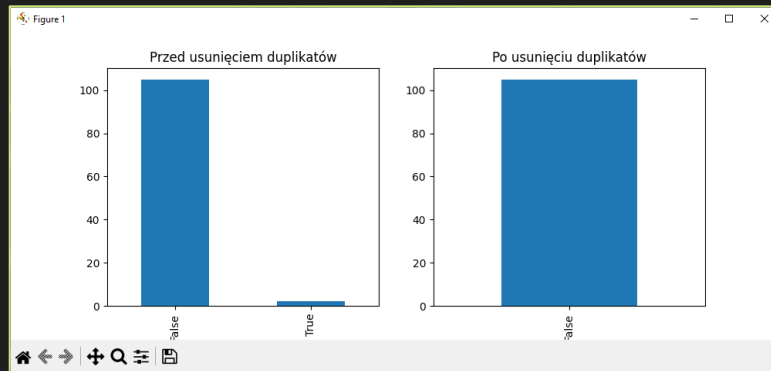
```
1 > import sys
2   import pandas as pd
3   import matplotlib.pyplot as plt
4
5 > def profile(dataframe, list_of_numeric_cols): ...
22
23 > def clean_missing_values(dataframe): ...
36
37 > def remove_duplicates(dataframe): ...
51
52 > def profiler_cleaner(file_path, *args): ...
78
79 # Przykład użycia: python data_profiler.py data.xlsx profile clean_missing_values
80 profiler_cleaner(sys.argv[1], *sys.argv[2:])
81
```

```
1 import sys
2 import pandas as pd
3 import matplotlib.pyplot as plt
4
5 def clean_missing_values(dataframe):
6     plt.figure(figsize=(10, 4))
7
8     print("Przed czyszczeniem: ", dataframe.isnull().sum())
9     plt.subplot(1, 2, 1)
10    plt.title("Przed czyszczeniem")
11    dataframe.isnull().sum().plot(kind='bar')
12
13    dataframe.dropna(inplace=True)
14    print("Po czyszczeniu: ", dataframe.isnull().sum())
15    plt.subplot(1, 2, 2)
16    plt.title("Po czyszczeniu")
17    dataframe.isnull().sum().plot(kind='bar')
18
19    plt.subplots_adjust(bottom=0.25)
20    plt.show()
21    return dataframe
22
```

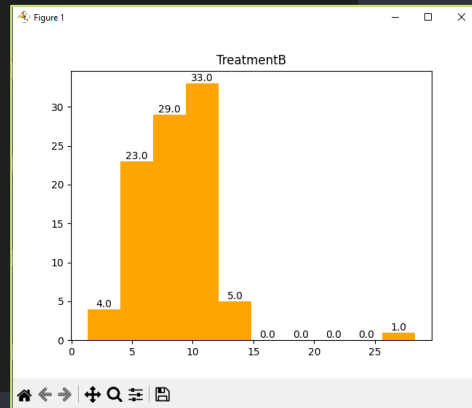
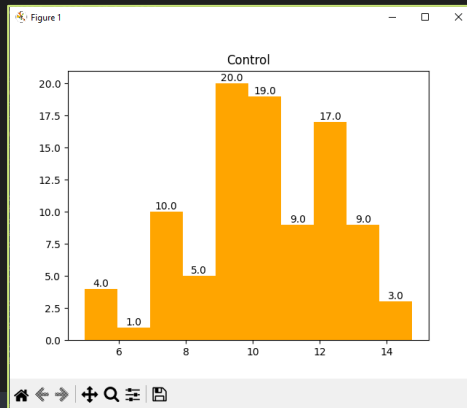


```
1 > import sys
2   import pandas as pd
3   import matplotlib.pyplot as plt
4
5 > def profile(dataframe, list_of_numeric_cols): ...
22
23 > def clean_missing_values(dataframe): ...
36
37 > def remove_duplicates(dataframe): ...
51
52 > def profiler_cleaner(file_path, *args): ...
78
79 # Przykład użycia: python data_profiler.py data.xlsx profile clean_missing_values
80 profiler_cleaner(sys.argv[1], *sys.argv[2:])
81
```

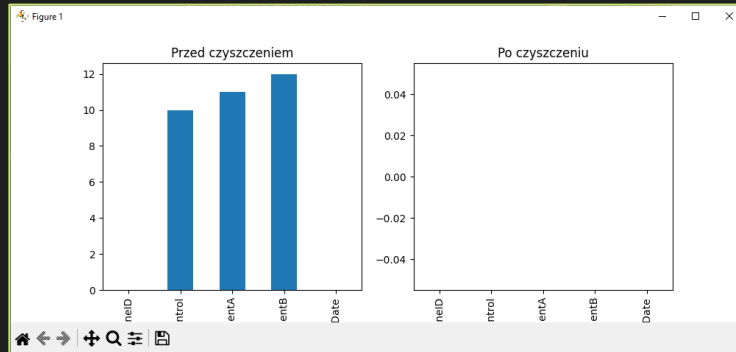
```
1 import sys
2 import pandas as pd
3 import matplotlib.pyplot as plt
4
5 def remove_duplicates(dataframe):
6     plt.figure(figsize=(10, 4))
7
8     print("Przed usunięciem duplikatów: ", dataframe.duplicated().sum())
9     plt.subplot(1, 2, 1)
10    plt.title("Przed usunięciem duplikatów")
11    dataframe.duplicated().value_counts().plot(kind='bar')
12
13    dataframe.drop_duplicates(inplace=True)
14    print("Po usunięciu duplikatów: ", dataframe.duplicated().sum())
15    plt.subplot(1, 2, 2)
16    plt.title("Po usunięciu duplikatów")
17    dataframe.duplicated().value_counts().plot(kind='bar')
18    plt.show()
19    return dataframe
20
```



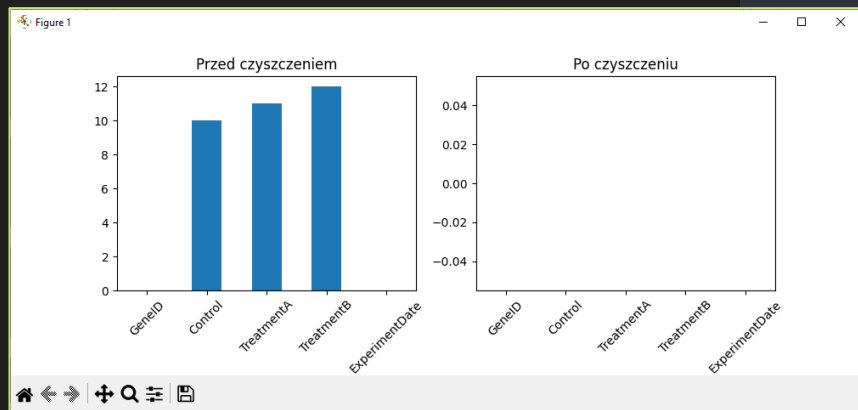
```
5 def profile(dataframe, list_of_numeric_cols):
6     print("HEAD AND TAIL:")
7     print(dataframe.head())
8     print(dataframe.tail())
9     print("INFO:")
10    print(dataframe.info())
11    print("DESCRIBE:")
12    print(dataframe.describe())
13    # Wstępna wizualizacja
14    for num_col in list_of_numeric_cols:
15        # values - wartości w każdym z przedziałów
16        # bins - granice przedziałów
17        # bars - słupki histogramu
18        values, bins, bars = plt.hist(dataframe[num_col], color = 'orange')
19        plt.title(num_col)
20        # Dodanie etykiet do słupków
21        plt.bar_label(bars, values)
22        plt.show()
23
```



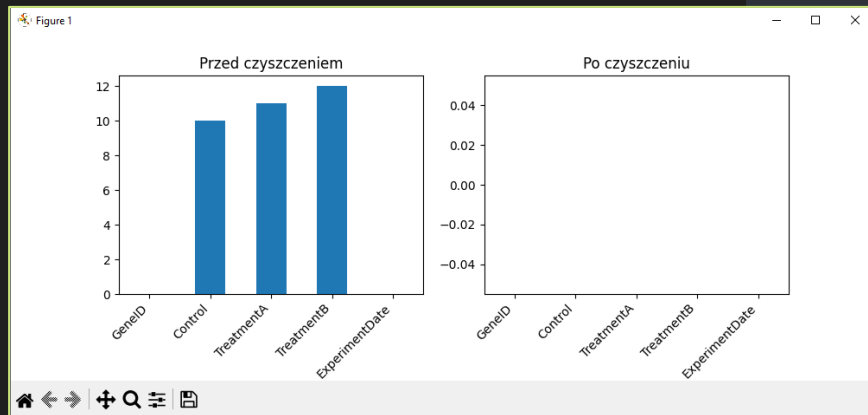

```
1 import sys
2 import pandas as pd
3 import matplotlib.pyplot as plt
4
5 def clean_missing_values(dataframe):
6     plt.figure(figsize=(10, 4))
7
8     print("Przed czyszczeniem: ", dataframe.isnull().sum())
9     plt.subplot(1, 2, 1)
10    plt.title("Przed czyszczeniem")
11    dataframe.isnull().sum().plot(kind='bar')
12
13    dataframe.dropna(inplace=True)
14    print("Po czyszczeniu: ", dataframe.isnull().sum())
15    plt.subplot(1, 2, 2)
16    plt.title("Po czyszczeniu")
17    dataframe.isnull().sum().plot(kind='bar')
18
19    plt.subplots_adjust(bottom=0.25)
20    plt.show()
21    return dataframe
22
```



```
1  import sys
2  import pandas as pd
3  import matplotlib.pyplot as plt
4
5  def clean_missing_values(dataframe):
6      plt.figure(figsize=(10, 4))
7
8      print("Przed czyszczeniem: ", dataframe.isnull().sum())
9      plt.subplot(1, 2, 1)
10     plt.title("Przed czyszczeniem")
11     dataframe.isnull().sum().plot(kind='bar')
12     plt.xticks(rotation=45)
13
14     dataframe.dropna(inplace=True)
15     print("Po czyszczeniu: ", dataframe.isnull().sum())
16     plt.subplot(1, 2, 2)
17     plt.title("Po czyszczeniu")
18     dataframe.isnull().sum().plot(kind='bar')
19     plt.xticks(rotation=45)
20
21     plt.subplots_adjust(bottom=0.25)
22     plt.show()
23     return dataframe
24
```



```
1 import sys
2 import pandas as pd
3 import matplotlib.pyplot as plt
4
5 def clean_missing_values(dataframe):
6     plt.figure(figsize=(10, 4))
7
8     print("Przed czyszczeniem: ", dataframe.isnull().sum())
9     plt.subplot(1, 2, 1)
10    plt.title("Przed czyszczeniem")
11    dataframe.isnull().sum().plot(kind='bar')
12    plt.xticks(rotation=45, ha='right')
13
14    dataframe.dropna(inplace=True)
15    print("Po czyszczeniu: ", dataframe.isnull().sum())
16    plt.subplot(1, 2, 2)
17    plt.title("Po czyszczeniu")
18    dataframe.isnull().sum().plot(kind='bar')
19    plt.xticks(rotation=45, ha='right')
20
21    plt.subplots_adjust(bottom=0.25)
22    plt.show()
23    return dataframe
24
```



```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # Ładujemy dane z pliku CSV
5 healthcare_data = pd.read_csv('healthcare_dataset.csv')
6
7 # Podgląd kolumn i typów danych
8 print(healthcare_data.info())
9
10 # Grupujemy wg. 'Medical Condition' i liczymy średni wiek dla każdej z grup
11 print(healthcare_data.groupby('Medical Condition')['Age'].mean())
12
13 # Liczymy liczbę wystąpień każdego typu krwi
14 print(healthcare_data['Blood Type'].value_counts())
15
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 15 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Name                  10000 non-null  object
1   Age                   10000 non-null  int64
2   Gender                10000 non-null  object
3   Blood Type            10000 non-null  object
4   Medical Condition     10000 non-null  object
5   Date of Admission     10000 non-null  object
6   Doctor                10000 non-null  object
7   Hospital              10000 non-null  object
8   Insurance Provider    10000 non-null  object
9   Billing Amount         10000 non-null  float64
10  Room Number           10000 non-null  int64
11  Admission Type        10000 non-null  object
12  Discharge Date        10000 non-null  object
13  Medication             10000 non-null  object
14  Test Results          10000 non-null  object
dtypes: float64(1), int64(2), object(12)
memory usage: 1.1+ MB
None
```

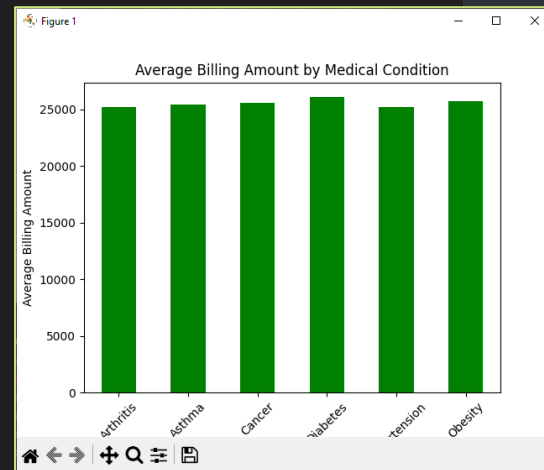
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # Ładujemy dane z pliku CSV
5 healthcare_data = pd.read_csv('healthcare_dataset.csv')
6
7 # Podgląd kolumn i typów danych
8 print(healthcare_data.info())
9
10 # Grupujemy wg. 'Medical Condition' i liczymy średni wiek dla każdej z grup
11 print(healthcare_data.groupby('Medical Condition')['Age'].mean())
12
13 # Liczymy liczbę wystąpień każdego typu krwi
14 print(healthcare_data['Blood Type'].value_counts())
15
```

Medical Condition	
Arthritis	51.530909
Asthma	51.445550
Cancer	51.583676
Diabetes	51.802218
Hypertension	50.737559
Obesity	51.633907
Name: Age, dtype: float64	

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # Ładujemy dane z pliku CSV
5 healthcare_data = pd.read_csv('healthcare_dataset.csv')
6
7 # Podgląd kolumn i typów danych
8 print(healthcare_data.info())
9
10 # Grupujemy wg. 'Medical Condition' i liczymy średni wiek dla każdej z grup
11 print(healthcare_data.groupby('Medical Condition')['Age'].mean())
12
13 # Liczymy liczbę wystąpień każdego typu krwi
14 print(healthcare_data['Blood Type'].value_counts())
15
```

```
AB-    1275
AB+    1258
B-     1252
O+     1248
O-     1244
B+     1244
A+     1241
A-     1238
Name: Blood Type, dtype: int64
```

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # Ładujemy dane z pliku CSV
5 healthcare_data = pd.read_csv('healthcare_dataset.csv')
6
7 # Podgląd kolumn i typów danych
8 print(healthcare_data.info())
9
10 # Grupujemy wg. 'Medical Condition' i liczymy średni wiek dla każdej z grup
11 print(healthcare_data.groupby('Medical Condition')['Age'].mean())
12
13 # Liczymy liczbę wystąpień każdego typu krwi
14 print(healthcare_data['Blood Type'].value_counts())
15
16 # Grupujemy wg. 'Blood Type' i liczymy średni wiek dla każdej z grup
17 avg_billing = healthcare_data.groupby('Medical Condition')['Billing Amount'].mean()
18
19 # Wykres słupkowy średniej kwoty rozliczenia dla każdej z grup
20 avg_billing.plot(kind='bar', color='green')
21 plt.title('Average Billing Amount by Medical Condition')
22 plt.xlabel('Medical Condition')
23 plt.ylabel('Average Billing Amount')
24 plt.xticks(rotation=45)
25 plt.show()
```



```

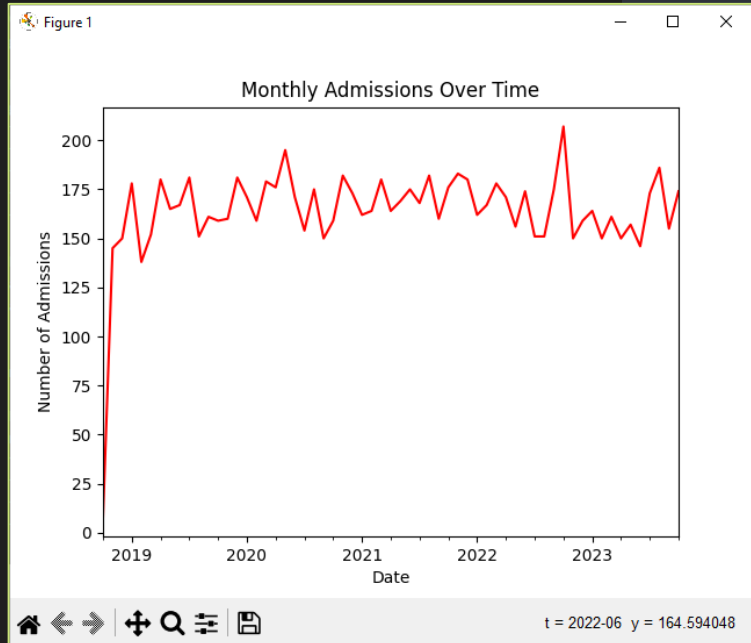
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # Ładujemy dane z pliku CSV
5 healthcare_data = pd.read_csv('healthcare_dataset.csv')
6
7 # Podgląd kolumn i typów danych
8 print(healthcare_data.info())
9
10 # Grupujemy wg. 'Medical Condition' i liczymy średni wiek dla
11 print(healthcare_data.groupby('Medical Condition')['Age'].mean())
12
13 # Liczymy liczbę wystąpień każdego typu krwi
14 print(healthcare_data['Blood Type'].value_counts())
15
16 # Grupujemy wg. 'Blood Type' i liczymy średni wiek dla każdej
17 avg_billing = healthcare_data.groupby('Medical Condition')['B
18
19 # Wykres słupkowy średniej kwoty rozliczenia dla każdej z gru
20 avg_billing.plot(kind='bar', color='green')
21 plt.title('Average Billing Amount by Medical Condition')
22 plt.xlabel('Medical Condition')
23 plt.ylabel('Average Billing Amount')
24 plt.xticks(rotation=45)
25 plt.show()
26
27 # Konwersja kolumny 'Date of Admission' na typ daty
28 healthcare_data['Date of Admission'] = pd.to_datetime(healthcare_data['Date of Admission'])
29 healthcare_data.set_index('Date of Admission', inplace=True)
30
31 # Liczymy liczbę przyjęć w każdym miesiącu
32 monthly_admissions = healthcare_data.resample('M').size()
33
34 print(monthly_admissions)
35
36 # Wykres liniowy liczby przyjęć w każdym miesiącu
37 monthly_admissions.plot(kind='line', color='red')
38 plt.title('Monthly Admissions Over Time')
39 plt.xlabel('Date')
40 plt.ylabel('Number of Admissions')
41 plt.show()
42

```

```

Date of Admission
2018-10-31      8
2018-11-30     145
2018-12-31     150
2019-01-31     178
2019-02-28     138
...
2023-06-30     146
2023-07-31     173
2023-08-31     186
2023-09-30     155
2023-10-31     174
Freq: M, Length: 61, dtype:
int64

```





"a scientist, a python, and an intelligent robot working together to analyze a large dataset"
by DALL-E 3

openai_teorja.py

openai_api.py

"Klasyczny"
ChatGPT



Kontekst
(~8k tokenów)



openai_teorja.py

openai_api.py

"Klasyczny"
ChatGPT



Kontekst
(~8k tokenów)



openai_teorja.py

openai_api.py

"Klasyczny"
ChatGPT



Kontekst
(~8k tokenów)



openai_teorja.py

openai_api.py

"Klasyczny"
ChatGPT



Kontekst
(~8k tokenów)



openai_teorja.py

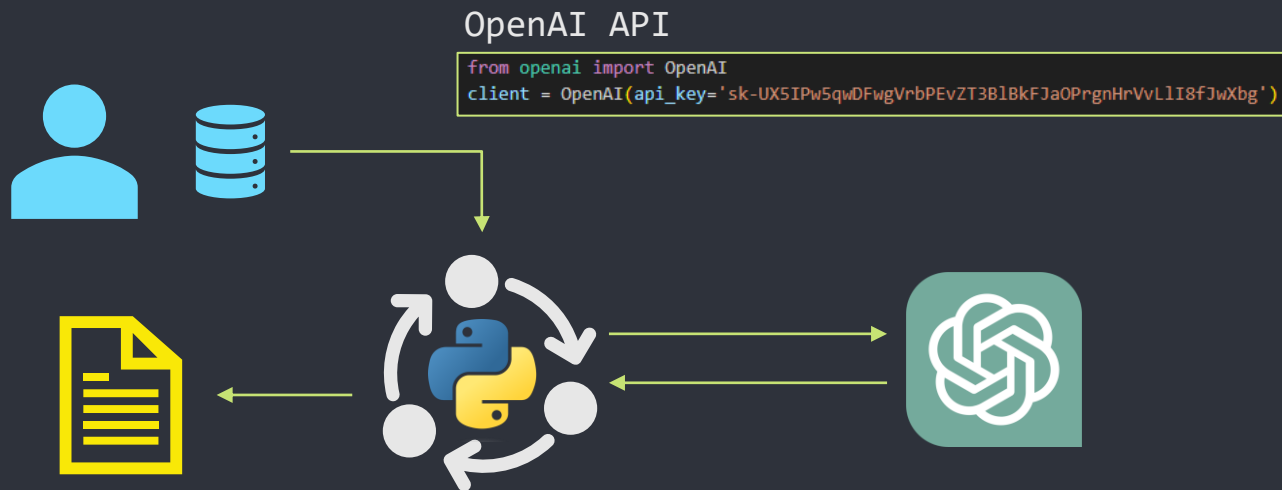
openai_api.py

"Klasyczny"
ChatGPT



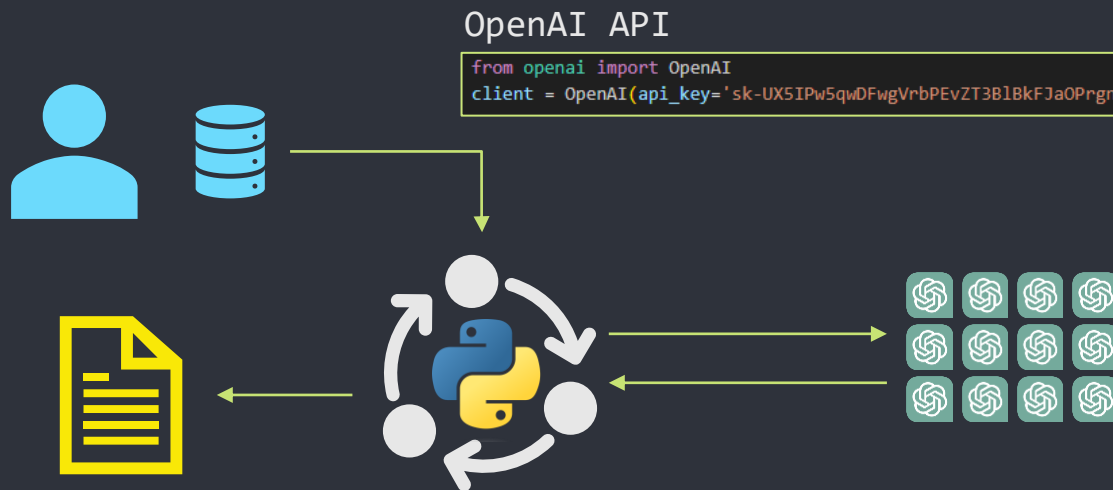
Kontekst
(~8k tokenów)





openai_teorja.py

openai_api.py



4. SZCZEGÓŁOWE DANE KLINICZNE

4.1 Wskazania do stosowania

- Bóle różnego pochodzenia o nasileniu małym do umiarkowanego (ból głowy, m.in. ból napięciowy i migrena, ból zębów, ból mięśni, stawów i kości, nerwobóle, bóle towarzyszące grypie i przeziębieniu).
- Gorączka różnego pochodzenia (m.in. w przebiegu grypy, przeziębienia lub innych chorób zakaźnych).
- Bolesne miesiączkowanie.

4.2 Dawkowanie i sposób podawania

Wyłącznie do podawania doustnego i do doraźnego stosowania.

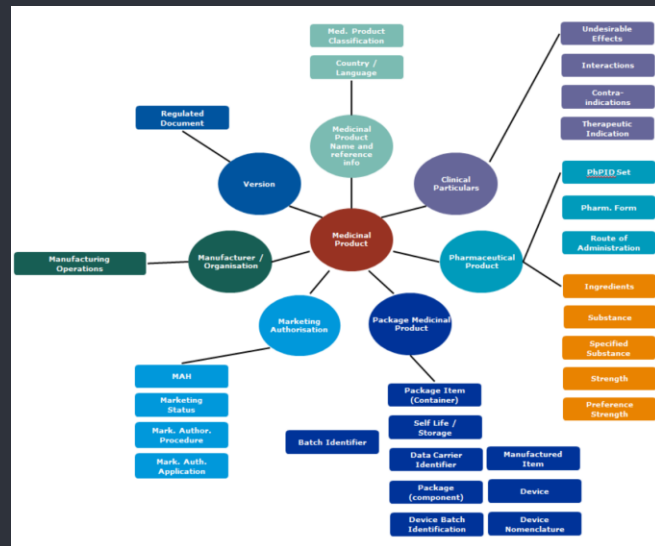
Dorośli i dzieci w wieku powyżej 12 lat: dawka początkowa to jedna tabletkę, następnie w razie potrzeby po 4 godzinach można przyjąć kolejną tabletkę. Nie stosować dawki większej niż trzy tabletki (1200 mg ibuprofenu) w ciągu doby. Należy zachować czterogodzinną przerwę pomiędzy dawkami. Tabletki należy popijać wodą.

Produktu leczniczego nie stosować u dzieci w wieku poniżej 12 lat.

U osób z dolegliwościami przewodu pokarmowego zaleca się przyjmowanie produktu leczniczego podczas posiłku.

W przypadku osób w podeszłym wieku modyfikacja dawki nie jest konieczna.

Działania niepożądane można ograniczyć, stosując najmniejszą skuteczną dawkę przez najkrótszy okres konieczny do złagodzenia objawów (patrz punkt 4.4). Jeżeli konieczne jest stosowanie produktu leczniczego dłużej niż przez 3 dni lub stan pacjenta pogarsza się, pacjent powinien skontaktować się z lekarzem.



źródło: EMA

`openai.py`

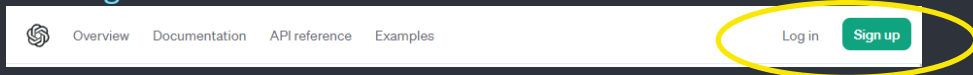
`api.py`



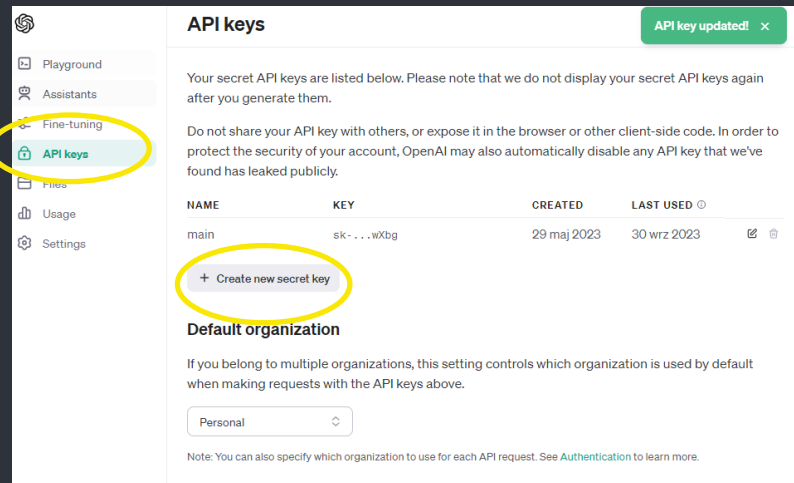
`platform.openai.com`

Jak zdobyć własny klucz API do OpenAI:

- Wchodzimy na platform.openai.com
- Tworzymy konto lub się logujemy za pomocą konta Google



- Wchodzimy w "API Keys" po lewej stronie
- Klikamy "Create new secret key"
- Wprowadzamy nazwę klucza (dowolną) i klikamy "Create secret key"
- Po wyświetleniu klucza na ekranie, kopiujemy w bezpieczne miejsce (możliwość podglądu klucza istnieje tylko tuż po jego utworzeniu)
- Klucz zawsze można usunąć i wygenerować nowy (w razie wycieku itp.)



- 1000 tokenów \approx 750 słów \approx 4000 znaków (j. angielski)
- nowi użytkownicy dostają \$5 (3 miesiące) na eksperymenty

MODEL	KONTEKST	KOSZT (INPUT)	KOSZT (OUTPUT)
gpt-3.5-turbo	16k tokenów	\$0.001/1k	\$0.002/1k
gpt-4	8k tokenów	\$0.03/1k	\$0.06/1k
gpt-4-32k	32k tokenów	\$0.06/1k	\$0.12/1k
gpt-4-turbo	128k tokenów	\$0.01/1k	\$0.03/1k

```
1  from openai import OpenAI
2  # wprowadzamy klucz API z platform.openai.com
3  client = OpenAI(api_key='sk-5VByl896AjlPeFAQMY9T3B1bkFJzBr9QmWyx3WCJDyDc1wA')
4
5  slowo = "Kot"
6  jezyk = "Angielski"
7
8  # komunikujemy się z modelem gpt-3.5-turbo za pomocą openai api
9  response = client.chat.completions.create(
10      model="gpt-3.5-turbo",
11      messages=[
12          {"role": "system", "content": "Jesteś asystentem-tłumaczem."},
13          {"role": "user", "content": f"Przetłumacz słowo {slowo} na język {jezyk}."},
14      ]
15  )
16  # wyciągamy treść odpowiedzi ze zwróconego obiektu
17  wynik = response.choices[0].message.content
18
19  print(wynik)
20
```

The translation of the word "Kot" in English is "Cat".

```
1 from openai import OpenAI
2 # wprowadzamy klucz API z platform.openai.com
3 client = OpenAI(api_key='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')
4
5 slowo = "Kot"
6 jezyk = "Angielski"
7
8 do_przetlumaczenia = [("kot", "angielski"), ("pies","francuski"), ("ptak","hiszpański")]
9
10 for slowo, jezyk in do_przetlumaczenia:
11     # komunikujemy się z modelem gpt-3.5-turbo za pomocą openai api
12     prompt = f"""Przetłumacz słowo {slowo} na język {jezyk}.
13     Wypisz tylko pojedyncze, wynikowe słowo, bez żadnego dodatkowego tekstu."""
14     response = client.chat.completions.create(
15         model="gpt-3.5-turbo",
16         messages=[
17             {"role": "system", "content": "Jesteś asystentem-tłumaczem."},
18             {"role": "user", "content": prompt},
19         ]
20     )
21     # wyciągamy treść odpowiedzi ze zwróconego obiektu
22     wynik = response.choices[0].message.content
23
24     print(wynik)
25
```

```
cat
chien
pájaro
```

"Na czym polega metoda PCR i jakie są jej etapy?"



	A	B
1	Nazwisko	Odpowiedz
2	Jan Kowalski	PCR, czyli reakcja łańcuchowa polimerazy, to metoda umożliwiająca szybkie i dokładne kopiowanie określonych fragmentów DNA. Proces ten składa się z trzech głównych etapów: denaturacji, gdzie podwójna helisa DNA jest rozdzielana przez podgrzewanie; hybrydyzacji, gdzie krótkie startery (primery) wiążą się z odpowiadającymi im sekwencjami na matrycy DNA; i elongacji, gdzie polimeraza DNA syntetyzuje nową nić DNA, komplementarną do matrycy. Metoda PCR znajduje zastosowanie w wielu dziedzinach, w tym w diagnostyce medycznej, badaniach genetycznych i kryminalistyce.
3	Anna Nowak	PCR to inaczej reakcja łańcuchowa polimerazy, służąca do amplifikacji fragmentów DNA (czyli ich mnożenia). Niestety nie pamiętam etapów reakcji.
4	Piotr Wiśniewski	Reakcja łańcuchowa polimerazy (PCR) to technika, która pozwala na wielokrotne powielenie wybranych fragmentów DNA. W pierwszym etapie, hybrydyzacji, łączymy primery z sekwencją DNA, którą chcemy powielić. Następnie następuje elongacja, która dobudowuje nukleotydy do łańcucha DNA.
5	Maria Jankowska	PCR to technika amplifikacji DNA, która pozwala na powielanie specyficznych sekwencji DNA. Proces rozpoczyna się od denaturacji, podczas której DNA jest podgrzewane, by rozdzielić obie nici. Następnie, w etapie hybrydyzacji, temperaturę obniża się, aby umożliwić przyłączenie primerów do matrycy DNA. W ostatniej fazie, elongacji, polimeraza DNA dodaje nukleotydy do powstającego łańcucha DNA. Metoda PCR jest szeroko stosowana w biologii molekularnej do klonowania DNA, sekwencjonowania i identyfikacji patogenów.
6	Krzysztof Nowicki	W metodzie PCR, najpierw dochodzi do rozplcenia podwójnej helisy DNA poprzez podniesienie temperatury (denaturacja). Następnie, w procesie hybrydyzacji, temperatura jest obniżana, co pozwala na przyłączenie się specyficznych dla danej sekwencji primerów. W trzecim kroku, elongacji, polimeraza DNA syntetyzuje nową nić DNA, rozpoczynając od primerów. PCR jest używany do amplifikacji DNA, co jest kluczowe w badaniach genetycznych, medycynie sądowej, i diagnostyce chorób.
7	Barbara Wójcik	Niestety nie mam pojęcia.
8	Andrzej Kaczmarek	PCR, czyli reakcja łańcuchowa polimerazy, jest techniką, która polega na wykorzystaniu światła UV do identyfikacji i selekcji specyficznych sekwencji DNA. W pierwszym kroku, DNA jest wystawiane na działanie światła UV, co powoduje fragmentację na mniejsze kawałki. Następnie, za pomocą specjalnych barwników, wybiera się te fragmenty DNA, które są potrzebne do dalszych badań. Ta metoda jest powszechnie używana w laboratoriach do szybkiej identyfikacji mutacji i chorób genetycznych.
9	Magdalena Wojciechowska	PCR to reakcja polimerazowa łańcucha. Służy do amplifikacji DNA. Składa się z 3 etapów: denaturacji, hybrydyzacji, oraz elongacji.
	Tomasz Adamczak	fragmentów DNA. W pierwszym etapie, denaturacji, podgrzewamy DNA, aby rozdzielić nici. W etapie hybrydyzacji, primer specyficzny dla danej sekwencji DNA wiąże się z nią, gdy temperatura jest obniżona. Następnie, w fazie elongacji, polimeraza DNA wydłuża nowe nici DNA, zaczynając od

```
1 from openai import OpenAI
2 import pandas as pd
3
4 data = pd.read_excel('odpowiedzi_studentow_egzamin_PCR.xlsx')
5
6 client = OpenAI(api_key='sk-UX5IPw5qWDFwgVrbPEvZT3B1bkFJaOPrgnHrVvLLI8fJwXbg')
7
8 pytanie = "Na czym polega metoda PCR i jakie są jej etapy?"
9
10 > pcr_opis = """Reakcja PCR (polimerazowej reakcji łańcuchowej, z ang. Polymerase Chain Reaction)
    amplifikacji określonego fragmentu DNA. Metoda ta umożliwia znaczne zwiększenie ilości konkretnego
    fragmentu DNA, co jest niezbędne do przeprowadzenia dalszych badań, takich jak sekwencjowanie,
    diagnostyka medyczna, kryminalistyka i inne dziedziny. ...
11
12
13
14
15
16
17
18 > instrukcje = f"""Dostaniesz pytanie egzaminacyjne, oraz odpowiedź jednego studenta. Dostaniesz r
19
20
21
22
23
24
25
26
27
28
29
30
31 lista_wynikow = []
32
33 for index, row in data.iterrows():
34     nazwisko = row['Nazwisko']
35     odpowiedz = row['Odpowiedz']
36     prompt = f"{instrukcje}\nOdpowiedź studenta: {odpowiedz}"
37     response = client.chat.completions.create(
38         model="gpt-3.5-turbo",
39         messages=[
40             {"role": "system", "content": "Jesteś asystentem odpowiedzialnym za sprawdzanie egzaminów"},
41             {"role": "user", "content": prompt},
42         ]
43     )
44     wynik = response.choices[0].message.content
45     lista_wynikow.append(wynik)
46
47 for w in lista_wynikow:
48     print(w)
49
```



```

1 from openai import OpenAI
2 import pandas as pd
3
4 data = pd.read_excel('odpowiedzi.xlsx')
5
6 client = OpenAI(api_key='sk-UXSI...')
7
8 pytanie = "Na czym polega metoda PCR?"
9
10 > pcr_opis = """Reakcja PCR (polimorfizm
    amplifikacji określonego fragmen
    diagnostyce medycznej, kryminali
11
12
13
14
15
16
17
18 > instrukcje = f"""Dostaniesz pyta
19
20
21 lista_wynikow = []
22
23 for index, row in data.iterrows():
24     nazwisko = row['Nazwisko']
25     odpowiedz = row['Odpowiedz']
26     prompt = f"{instrukcje}\nOdpowiedz na pytanie: {pytanie}"
27     response = client.chat.completions.create(
28         model="gpt-3.5-turbo",
29         messages=[
30             {"role": "system", "content": "Jesteś asystentem, który odpowiada na pytania dotyczące PCR."},
31             {"role": "user", "content": prompt}
32         ]
33     )
34     wynik = response.choices[0].message.content
35     lista_wynikow.append(wynik)
36
37 for w in lista_wynikow:
38     print(w)
39

```

[3, 3, "Student prawidłowo scharakteryzował metodę oraz podał wszystkie jej etapy."]

[0, 0, "Student nie rozumie zagadnienia."]

[3, 2, "Student poprawnie opisał metodę PCR i wymienił dwa z trzech etapów."]

[3, 3, "Student poprawnie opisał metodę PCR oraz wymienił i opisał wszystkie jej etapy."]

Prawidłowa odpowiedź. [3, 3, "Student prawidłowo scharakteryzował metodę oraz podał wszystkie jej etapy."]

[0, 0, "Student nie rozumie zagadnienia."]

[0, 0, "Odpowiedź studenta jest nieprawidłowa. Metoda PCR polega na amplifikacji określonego fragmentu DNA, a nie na identyfikacji i selekcji sekwencji DNA za pomocą światła UV. Student nie opisał żadnego z etapów metody."]

[3, 3, "Student prawidłowo scharakteryzował metodę oraz podał wszystkie jej etapy."]

[0, 0, "Student nie rozumie zagadnienia. Nie wymienił żadnego etapu ani nie opisał metody."]

[0, 0, "Odpowiedź studenta jest nieprawidłowa. Metoda PCR nie służy do badania struktur przestrzennych białek, ale do amplifikacji DNA."]

[0, 0, "Student nie rozumie zagadnienia metody PCR."]

[3, 3, "Student prawidłowo scharakteryzował metodę oraz podał wszystkie jej etapy."]

[3, 3, "Student prawidłowo scharakteryzował metodę oraz podał wszystkie jej etapy."]

```

31 lista_wynikow = []
32
33 max_liczba_prob = 3
34
35 for index, row in data.iterrows():
36     nazwisko = row['Nazwisko']
37     odpowiedz = row['Odpowiedz']
38     prompt = f"{instrukcje}\nOdpowiedz na pytanie: {nazwisko}
39     liczba_prob = 0
40     wynik_ok = False
41     while not wynik_ok:
42         liczba_prob += 1
43         print(f"Próba {liczba_prob} dla studenta: {nazwisko}")
44         response = client.chat.completions.create(
45             model="gpt-3.5-turbo",
46             messages=[
47                 {"role": "system", "content": "Jesteś asystentem, który odpowiada na pytania."},
48                 {"role": "user", "content": prompt}
49             ]
50         )
51         wynik = response.choices[0].text
52         print(wynik)
53         if isinstance(eval(wynik), list):
54             print("jest lista")
55             lista_wynikow.append(wynik)
56             wynik_ok = True
57         else:
58             if liczba_prob == max_liczba_prob:
59                 print(f"Nie udało się uzyskać odpowiedzi dla studenta: {nazwisko}")
60                 lista_wynikow.append(None)
61                 wynik_ok = True
62             else:
63                 continue
64 df = pd.DataFrame(lista_wynikow, index=range(len(lista_wynikow)))
65 df.to_excel('lista_wynikow.xlsx', index=False)
66

```

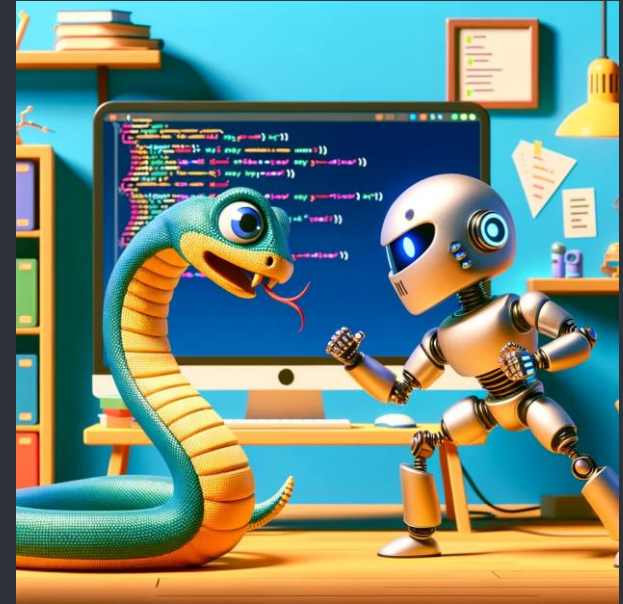
```

Próba 1 dla studenta: Jan Kowalski
jest lista
Próba 1 dla studenta: Anna Nowak
jest lista
Próba 1 dla studenta: Piotr Wiśniewski
jest lista
Próba 1 dla studenta: Maria Jankowska
jest lista
Próba 1 dla studenta: Krzysztof Nowicki
jest lista
Próba 1 dla studenta: Barbara Wójcik
Próba 2 dla studenta: Barbara Wójcik
jest lista
Próba 1 dla studenta: Andrzej Kaczmarek
jest lista
Próba 1 dla studenta: Magdalena Wojciechowska
jest lista
Próba 1 dla studenta: Tomasz Adamczak
jest lista
Próba 1 dla studenta: Ewa Piotrowska
jest lista
Próba 1 dla studenta: Michał Kowalczyk
jest lista
Próba 1 dla studenta: Paweł Michalski
jest lista
Próba 1 dla studenta: Grzegorz Krawczyk
jest lista

```

```
1 import PyPDF2
2 from openai import OpenAI
3 import pandas as pd
4 import os
5
6 klient = OpenAI(api_key='sk-UXSIPw5qw0FwgVrbPEvZT381bkfJaOPrgnHrVvLI18fjwXbg')
7
8 def pierwsza_strona_pdf_na_tekst(sciezka_pliku_pdf):
9     """
10     Czyta plik PDF i zwraca zawartość pierwszej strony jako string.
11     """
12     with open(sciezka_pliku_pdf, 'rb') as plik_pdf:
13         pdfreader = PyPDF2.PdfFileReader(plik_pdf)
14         strona = pdfreader.getPage(0)
15         tekst_strony = strona.extractText()
16
17     return tekst_strony
18
19 def uruchom_openai(prompt):
20     odpowiedz = klient.chat.completions.create(
21         model="gpt-3.5-turbo",
22         messages=[
23             {"role": "system", "content": "Jesteś pomocnym asystentem."},
24             {"role": "user", "content": prompt},
25         ]
26     )
27     return odpowiedz.choices[0].message.content
28
29 def zbuduj_prompt(tresc_pierwszej_strony):
30     """
31     Funkcja ma zwracać prompt dla modelu OpenAI, który będzie zawierał
32     wszelkie niezbędne instrukcje, oraz treść pierwszej strony pliku PDF.
33     """
34     pass
35
36 def uporządkuj_dane_z_pdfa(sciezka_do_pdf):
37     """
38     Funkcja, wewnątrz której odbywa się komunikacja z modelem OpenAI.
39     Funkcja ma zwracać listę danych wyczytanych z pierwszej strony
40     pliku PDF. lista ma zawierać: tytuł, listę nazwisk autorów,
41     oraz krótkie podsumowanie abstraktu.
42     """
43     pass
44
45 lista_pdfow = ['pdfy/' + f for f in os.listdir('pdfy')]
46 print(lista_pdfow)
47
48
```

PROBLEM	ROZWIĄZANIE
Halucynacje	Kontrola nad danymi
Limit wielkości kontekstu	Wektorowe bazy danych
Bezpieczeństwo danych	Azure OpenAI
Testowanie rozwiązań	Zautomatyzowane benchmarki
Struktura danych wyjściowych	Prompt engineering





PODSTAWY PROMPT ENGINEERING
JAK KOMUNIKOWAĆ SIĘ Z MODELAMI JĘZYKOWYMI ABY
WYKORZYSTAĆ ICH POTENCJAŁ W BADANIACH NAUKOWYCH

PROWADZI:
dr Mateusz Dobrychłop

Tytuł warsztatu: „**Podstawy prompt engineering – jak komunikować się z modelami językowymi, aby wykorzystać ich potencjał w badaniach naukowych**”

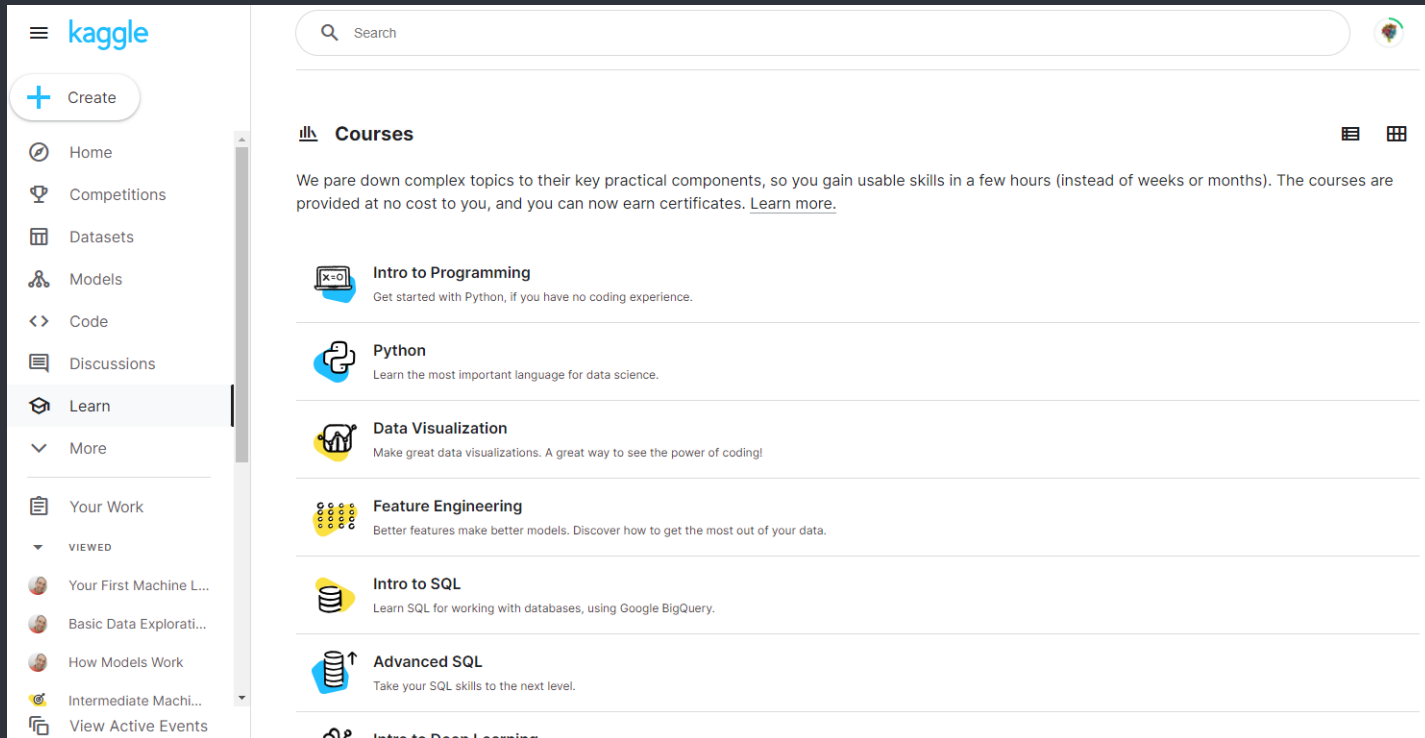
Prowadzący: dr Mateusz Dobrychłop

5, 12 i 19 marca 2024 roku

Godziny trwania: 17:00-19:30

Termin zgłoszenia w I turze: 14 grudnia 2023 roku

fundacja-tygiel.pl



The screenshot displays the Kaggle website's 'Learn' section. On the left is a navigation sidebar with options: Home, Competitions, Datasets, Models, Code, Discussions, Learn (highlighted), and More. Below these are 'Your Work' and a 'VIEWED' list containing 'Your First Machine L...', 'Basic Data Explorati...', 'How Models Work', 'Intermediate Machi...', and 'View Active Events'. The main content area features a search bar and a 'Courses' section. A paragraph explains that courses break down complex topics into practical components, available at no cost with certificates. Below this, a list of courses is shown: 'Intro to Programming' (Python for beginners), 'Python' (most important language for data science), 'Data Visualization' (making great visualizations), 'Feature Engineering' (better features for better models), 'Intro to SQL' (working with databases using Google BigQuery), 'Advanced SQL' (taking SQL skills to the next level), and 'Intro to Deep Learning'.

Courses

We pare down complex topics to their key practical components, so you gain usable skills in a few hours (instead of weeks or months). The courses are provided at no cost to you, and you can now earn certificates. [Learn more.](#)

- Intro to Programming**
Get started with Python, if you have no coding experience.
- Python**
Learn the most important language for data science.
- Data Visualization**
Make great data visualizations. A great way to see the power of coding!
- Feature Engineering**
Better features make better models. Discover how to get the most out of your data.
- Intro to SQL**
Learn SQL for working with databases, using Google BigQuery.
- Advanced SQL**
Take your SQL skills to the next level.
- Intro to Deep Learning**

Serdecznie dziękuję! = {



e-mail : ['mateusz.dobrychlop@gmail.com'],
linkedin : ['linkedin.com/in/mdobrychlop'],

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**