

```
1
2
3  Praktyczne_zastosowanie_Pythona = {
4
5      w_naukach : ['biologicznych', 'i medycznych'],
6      dla :      ['początkujących'],
7      część :    ['druga'],
8
9          # Mateusz Dobrychłop, 21 grudnia 2022
10
11
12  }
```

```
1 Plan_szkolenia = {
2
3     01 Absolutne podstawy
4       [ uruchamianie kodu, podstawowe
5         typy danych, warunki ]
6
7     02 Praca z sekwencjami
8       [ listy, pętle, pliki
9         tekstowe ]
10
11     03 Arkusze danych
12       [ pandas, arkusze,
13         dokumentacja ]
14 }
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14

```
Rozwiązania = {  
    'Zadań': 'domowych'  
}
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

`_03 = {`

`'Arkusze kalkulacyjne' :`

`[ 'słowniki',`

`'pandas',`

`'arkusze Excela',`

`'dokumentacja',`

`]`

`}`

# Instalujemy\_pandas = {

'Krok\_1': Otwieramy linię komend (np.  
PowerShell lub cmd)

'Krok\_2': Wpisujemy i zatwierdzamy komendy:  
**pip install pandas**  
**pip install openpyxl**

'Krok\_3': W nowym skrypcie Pythona, w  
pierwszej linii wpisujemy:  
**import pandas**

'Krok\_4': Jeżeli skrypt uruchamia się bez  
problemu, moduł został zainstalowany  
poprawnie

}

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Dobry> pip install pandas
Collecting pandas
  Downloading pandas-1.5.2-cp310-cp310-win_amd64.whl (10.4 MB)
    ----- 10.4/10.4 MB 18.2 MB/s eta 0:00:00
Collecting numpy>=1.21.0
  Downloading numpy-1.23.5-cp310-cp310-win_amd64.whl (14.6 MB)
    ----- 14.6/14.6 MB 24.3 MB/s eta 0:00:00
Collecting pytz>=2020.1
  Downloading pytz-2022.6-py2.py3-none-any.whl (498 kB)
    ----- 498.1/498.1 kB 15.7 MB/s eta 0:00:00
Collecting python-dateutil>=2.8.1
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
    ----- 247.7/247.7 kB 2.5 MB/s eta 0:00:00
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: pytz, six, numpy, python-dateutil, pandas
Successfully installed numpy-1.23.5 pandas-1.5.2 python-dateutil-2.8.2 pytz-2022.6 six-1.16.0


[notice] A new release of pip available: 22.2.1 -> 22.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
PS C:\Users\Dobry>
```


# Rozwiazanie\_alternatywne = {

'Krok\_1': Wchodzimy na stronę:  
<https://www.onlinegdb.com/>

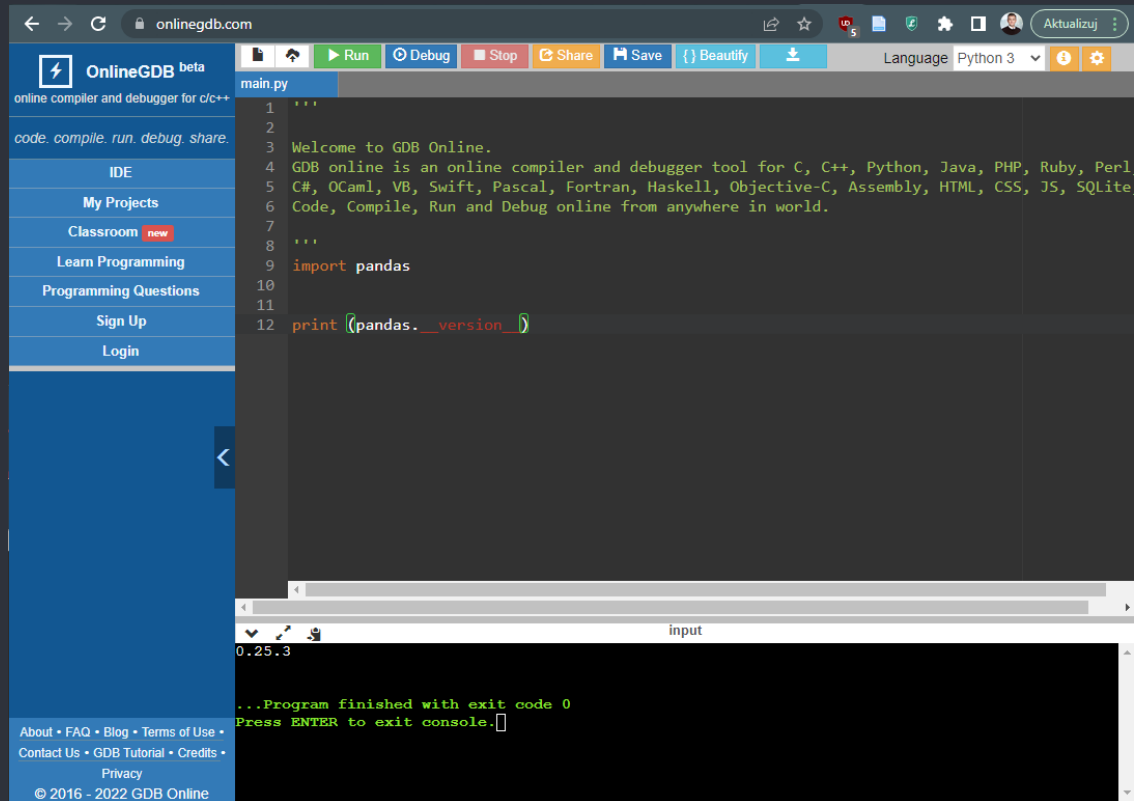
'Krok\_2': Z listy „Language” (prawy górny  
róg) wybieramy „Python 3”

Language Python 3 ▾

'Krok\_3': Przesyłamy pliki wejściowe za  
pomocą ikonki  (lewy górny róg)

'Krok\_4': Wykonujemy ćwiczenia w przeglądarkowym  
edytorze. Uruchamiamy kod za pomocą  
przycisku  (lewy górny róg)

}



The screenshot shows the OnlineGDB web interface. The browser address bar displays 'onlinegdb.com'. The interface includes a sidebar on the left with navigation links: IDE, My Projects, Classroom (marked 'new'), Learn Programming, Programming Questions, Sign Up, and Login. The main editor area shows a file named 'main.py' with the following Python code:

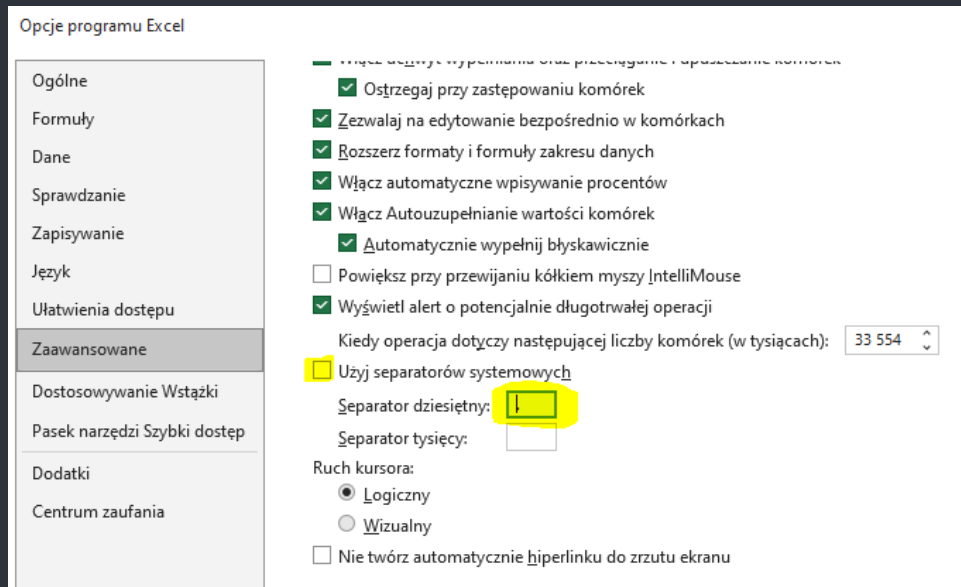
```
1 '''
2
3 Welcome to GDB Online.
4 GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, Ruby, Perl
5 C#, OCaml, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS, SQLite
6 Code, Compile, Run and Debug online from anywhere in world.
7
8 '''
9 import pandas
10
11
12 print(pandas.__version__)
```

Below the code editor, the 'input' section shows the output of the program: '0.25.3'. At the bottom, a message states: '...Program finished with exit code 0 Press ENTER to exit console.'



```
1  Wazne_uwagi = {
2
3      '_1': 'Prawidłowo zainstalowany pandas pozwala na bezpośrednią pracę
4            z plikami.xlsx.'
5
6      '_2': 'OnlineGDB nie radzi sobie z nimi, więc osoby
7            korzystające z niego, będą najpierw musiały
8            przekonwertować plik.xlsx na plik.csv'
9
10     '_3': 'Praca na plikach.xlsx i.csv z wykorzystaniem pandas
11           wygląda prawie identycznie'
12
13     '_4': 'Konwersja do.csv i inne przydatne instrukcje -
14           na kolejnych slajdach'
15 }
```

```
1 # Jeżeli konieczne jest skorzystanie z OnlineGDB, należy najpierw zapisać wejściowy
2 # arkusz w formacie CSV (comma separated values). Najpierw jednak warto zmienić separator
3 # dziesiętny w opcjach Excela na kropkę, żeby Python odpowiednio interpretował odczytane
4 # liczby zmiennoprzecinkowe.
```



```
1 # Następnie plik należy zapisać jako plik CSV (Plik CSV UTF-8 (rozdzielany przecinkami).
```



|    | A     | B     | C      | D             | E  |
|----|-------|-------|--------|---------------|----|
| 1  | group | water | eggs   | fertilisation |    |
| 2  |       | 1     | 0,0638 | 6,4342        | 97 |
| 3  |       | 1     | 0,0635 | 6,4322        | 97 |
| 4  |       | 1     | 0,0637 | 6,4298        | 97 |
| 5  |       | 1     | 0,0645 | 11,483        | 97 |
| 6  |       | 1     | 0,0652 | 11,4875       | 97 |
| 7  |       | 1     | 0,0651 | 11,4819       | 97 |
| 8  |       | 1     | 0,0649 | 24,7564       | 97 |
| 9  |       | 1     | 0,0648 | 24,7695       | 97 |
| 10 |       | 1     | 0,0646 | 24,7513       | 97 |
| 11 |       | 1     | 0,0639 | 20,3298       | 97 |
| 12 |       | 1     | 0,0638 | 20,3325       | 97 |

f\_data

```
1  # Kiedy iterujemy pętlą for po liście mniejszych sekwencji, to możemy bezpośrednio
2  # korzystać z elementów tych podrzędnych sekwencji, podając pętli for więcej niż
3  # jedną zmienną tuż po słowie kluczowym for.
4
5  people = [['Mary', 33], ['John', 25], ['Anna', 56]]
6
7  for p in people:
8      print(a)
9
10 for a, b in people:
11     print('name:', a, ', age:', b)
12
13
14
15
16
17
18
19
20
21
22
23
24
25
```

```
['Mary', 33]
['John', 25]
['Anna', 56]
name: Mary , age: 33
name: John , age: 25
name: Anna , age: 56
```

```
1  # Słownik jest bardzo przydatnym typem danych, który pozwala nam uporządkować
2  # dane w parach klucz:wartość. Klucze w słowniku muszą być unikalne, wartości
3  # niekoniecznie.
4
5  people = {'Mary':33, 'John':25, 'Anna':56}
6
7  print(people['John'])
8
9  for key in people:
10     print(key, people[key])
11
12  people['Paul'] = 45
13
14  print('Po dodaniu Paula:')
15
16  for key in people:
17     print(key, people[key])
18
19
20
21
22
23
24
25
```

```
25
Mary 33
John 25
Anna 56
Po dodaniu Paula:
Mary 33
John 25
Anna 56
Paul 45
```

```
1  # Słownik jest bardzo przydatnym typem danych, który pozwala nam uporządkować
2  # dane w parach klucz:wartość. Klucze w słowniku muszą być unikalne, wartości
3  # niekoniecznie.
4
5  sequence = 'GGCCCAUCGUCUAGCGGUUAGGACG'
6
7  comp_dict = {'A':'U', 'G':'C', 'U':'A', 'C':'G'}
8
9  comp_sequence = ''
10
11  for s in sequence:
12      comp_sequence += comp_dict[s]
13
14  print(sequence)
15  print(comp_sequence)
16
17
18
19
20
21
22
23
24
25
```

```
GGCCCAUCGUCUAGCGGUUAGGACG
CCGGGGUAGCAGAUCGCCAAUCCUGC
```

```
1  # Dokończ kod poniżej. Kod ma wykorzystywać pętlę for tak, żeby dane z listy people
2  # znalazły się w słowniku out_dictionary.
3  # Następnie zaimplementuj pętlę while, która przyjmie od użytkownika dwa kolejne
4  # imiona i dwie kolejne liczby i doda te dane do słownika (pierwsze imię : pierwsza liczba,
5  # drugie imię : druga liczba). Wprowadź zabezpieczenie, uniemożliwiające wprowadzenie
6  # liczby większej niż 150 (niech program pyta o liczbę aż nie zostanie
7  # mniejszej lub równej 150).
8  # Na końcu wypisz słownik.
9
10 people = [['Mary', 33], ['John', 25], ['Anna', 56]]
11
12 out_dictionary = {}
13
14
15
16
17
18
19
20
21
22
23
24
25
```



```
1  # Pandas to biblioteka służąca do analizy danych, pozwalająca na przetwarzanie
2  # danych w strukturach zwanych Data Frames.
3  # W Pythonie część funkcji nie jest domyślnie dostępna do wykorzystania w kodzie
4  # - czasami, żeby wykorzystać funkcję, należy ją zaimportować (lub importować
5  # cały moduł, którego jest częścią). Służy do tego instrukcja import.
6  # Wiele modułów dostępnych do zaimportowania jest instalowana razem z Pythonem
7  # w systemie. Niektóre moduły, takie jak (obecnie) pandas, wymagają dodatkowej
8  # instalacji.
9
10 import pandas
11
12 df = pandas.read_excel('fertilisation.xlsx', 'f_data') # Data Frame
13
14 print(df)
15
16
17
18
19
20
21
22
23
24
25
```

|    | group | water  | eggs    | fertilisation |
|----|-------|--------|---------|---------------|
| 0  | 1     | 0.0638 | 6.4342  | 97            |
| 1  | 1     | 0.0635 | 6.4322  | 97            |
| 2  | 1     | 0.0637 | 6.4298  | 97            |
| 3  | 1     | 0.0645 | 11.4830 | 97            |
| 4  | 1     | 0.0652 | 11.4875 | 97            |
| 5  | 1     | 0.0651 | 11.4819 | 97            |
| 6  | 1     | 0.0649 | 24.7564 | 97            |
| 7  | 1     | 0.0648 | 24.7695 | 97            |
| 8  | 1     | 0.0646 | 24.7513 | 97            |
| 9  | 1     | 0.0639 | 20.3298 | 97            |
| 10 | 1     | 0.0638 | 20.3325 | 97            |
| 11 | 1     | 0.0637 | 20.3271 | 97            |

```
1  # Pandas to biblioteka służąca do analizy danych, pozwalająca na przetwarzanie
2  # danych w strukturach zwanych Data Frames.
3  # W Pythonie część funkcji nie jest domyślnie dostępna do wykorzystania w kodzie
4  # - czasami, żeby wykorzystać funkcję, należy ją zaimportować (lub importować
5  # cały moduł, którego jest częścią). Służy do tego instrukcja import.
6  # Wiele modułów dostępnych do zaimportowania jest instalowana razem z Pythonem
7  # w systemie. Niektóre moduły, takie jak (obecnie) pandas, wymagają dodatkowej
8  # instalacji.
9
10 import pandas as pd
11
12 df = pd.read_excel('fertilisation.xlsx', 'f_data')
13
14 print(df)
```

|    | group | water  | eggs    | fertilisation |
|----|-------|--------|---------|---------------|
| 0  | 1     | 0.0638 | 6.4342  | 97            |
| 1  | 1     | 0.0635 | 6.4322  | 97            |
| 2  | 1     | 0.0637 | 6.4298  | 97            |
| 3  | 1     | 0.0645 | 11.4830 | 97            |
| 4  | 1     | 0.0652 | 11.4875 | 97            |
| 5  | 1     | 0.0651 | 11.4819 | 97            |
| 6  | 1     | 0.0649 | 24.7564 | 97            |
| 7  | 1     | 0.0648 | 24.7695 | 97            |
| 8  | 1     | 0.0646 | 24.7513 | 97            |
| 9  | 1     | 0.0639 | 20.3298 | 97            |
| 10 | 1     | 0.0638 | 20.3325 | 97            |
| 11 | 1     | 0.0637 | 20.3271 | 97            |

```
1  # Pandas to biblioteka służąca do analizy danych, pozwalająca na przetwarzanie
2  # danych w strukturach zwanych Data Frames.
3  # W Pythonie część funkcji nie jest domyślnie dostępna do wykorzystania w kodzie
4  # - czasami, żeby wykorzystać funkcję, należy ją zaimportować (lub importować
5  # cały moduł, którego jest częścią). Służy do tego instrukcja import.
6  # Wiele modułów dostępnych do zaimportowania jest instalowana razem z Pythonem
7  # w systemie. Niektóre moduły, takie jak (obecnie) pandas, wymagają dodatkowej
8  # instalacji.
9
10 from pandas import read_excel
11
12 df = read_excel('fertilisation.xlsx', 'f_data')
13
14 print(df)
```

|    | group | water  | eggs    | fertilisation |
|----|-------|--------|---------|---------------|
| 0  | 1     | 0.0638 | 6.4342  | 97            |
| 1  | 1     | 0.0635 | 6.4322  | 97            |
| 2  | 1     | 0.0637 | 6.4298  | 97            |
| 3  | 1     | 0.0645 | 11.4830 | 97            |
| 4  | 1     | 0.0652 | 11.4875 | 97            |
| 5  | 1     | 0.0651 | 11.4819 | 97            |
| 6  | 1     | 0.0649 | 24.7564 | 97            |
| 7  | 1     | 0.0648 | 24.7695 | 97            |
| 8  | 1     | 0.0646 | 24.7513 | 97            |
| 9  | 1     | 0.0639 | 20.3298 | 97            |
| 10 | 1     | 0.0638 | 20.3325 | 97            |
| 11 | 1     | 0.0637 | 20.3271 | 97            |

```
1  # Pandas to biblioteka służąca do analizy danych, pozwalająca na przetwarzanie
2  # danych w strukturach zwanych Data Frames.
3  # W Pythonie część funkcji nie jest domyślnie dostępna do wykorzystania w kodzie
4  # - czasami, żeby wykorzystać funkcję, należy ją zaimportować (lub importować
5  # cały moduł, którego jest częścią). Służy do tego instrukcja import.
6  # Wiele modułów dostępnych do zaimportowania jest instalowana razem z Pythonem
7  # w systemie. Niektóre moduły, takie jak (obecnie) pandas, wymagają dodatkowej
8  # instalacji.
9
10 import pandas
11
12 # DLA OSOB PRACUJACYCH NA PLIKACH XLSX:
13 # df = pandas.read_excel('fertilisation.xlsx', 'f_data')
14
15 # DLA OSOB PRACUJACYCH NA PLIKACH CSV:
16 df = pandas.read_csv('fertilisationcsv.csv', sep=';')
17
18 print(df)
19
20
21
22
23
24
25
```

| group | water | eggs   | fertilisation |      |
|-------|-------|--------|---------------|------|
| 0     | 1.0   | 0.0638 | 6.4342        | 97.0 |
| 1     | 1.0   | 0.0635 | 6.4322        | 97.0 |
| 2     | 1.0   | 0.0637 | 6.4298        | 97.0 |
| 3     | 1.0   | 0.0645 | 11.4830       | 97.0 |
| 4     | 1.0   | 0.0652 | 11.4875       | 97.0 |
| ..    | ...   | ...    | ...           | ...  |
| 57    | 4.0   | 0.1259 | 4.0909        | 2.0  |
| 58    | 4.0   | 0.1258 | 4.0911        | 2.0  |
| 59    | 4.0   | 0.1260 | 4.0912        | 2.0  |
| 60    | NaN   | NaN    | NaN           | NaN  |
| 61    | NaN   | NaN    | NaN           | NaN  |

[62 rows x 4 columns]

```
1  # Po wierszach struktury Data Frame można iterować, podobnie jak po liście.  
2  # Do konkretnych kolumn każdego wiersza odnosimy się z kolei podobnie jak do kluczy  
3  # w słowniku.  
4  
5  import pandas  
6  
7  df = pandas.read_excel('fertilisation.xlsx', 'f_data')  
8  # df = pandas.read_csv('fertilisationcsv.csv', sep=';')  
9  
10 for index, row in df.iterrows():  
11     print(row['eggs'])  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25
```

```
6.4342  
6.4322  
6.4298  
11.483  
11.4875  
11.4819  
24.7564  
24.7695  
24.7513  
20.3298  
20.3325  
20.3271
```

```
1  # Po wierszach struktury Data Frame można iterować, podobnie jak po liście.  
2  # Do konkretnych kolumn każdego wiersza odnosimy się z kolei podobnie jak do kluczy  
3  # w słowniku.  
4  
5  import pandas  
6  
7  df = pandas.read_excel('fertilisation.xlsx', 'f_data')  
8  # df = pandas.read_csv('fertilisationcsv.csv', sep=';')  
9  
10 for index, row in df.iterrows():  
11     eggs_value = row['eggs']  
12     if eggs_value > 20.0:  
13         print(eggs_value)  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25
```

```
24.7564  
24.7695  
24.7513  
20.3298  
20.3325  
20.3271  
52.4587  
52.4685  
52.4509  
46.8095  
46.8126  
46.7815
```

```
1  # Po wierszach struktury Data Frame można iterować, podobnie jak po liście.
2  # Do konkretnych kolumn każdego wiersza odnosimy się z kolei podobnie jak do kluczy
3  # w słowniku.
4
5  import pandas
6
7  df = pandas.read_excel('fertilisation.xlsx', 'f_data')
8  # df = pandas.read_csv('fertilisationcsv.csv', sep=';')
9
10 for index, row in df.iterrows():
11     if row['group'] == 2:
12         print(row['eggs'])
13
14
15
16
17
18
19
20
21
22
23
24
25
```

```
5.9069
5.9128
5.9045
16.5885
16.5963
16.5874
52.4587
52.4685
52.4509
10.5674
10.6612
10.4528
9.2451
9.2536
9.2455
```

```
1  # Po wierszach struktury Data Frame można iterować, podobnie jak po liście.
2  # Do konkretnych kolumn każdego wiersza odnosimy się z kolei podobnie jak do kluczy
3  # w słowniku.
4
5  import pandas
6
7  df = pandas.read_excel('fertilisation.xlsx', 'f_data')
8  # df = pandas.read_csv('fertilisationcsv.csv', sep=';')
9
10 group_sum = 0.0
11 group_count = 0
12
13 for index, row in df.iterrows():
14     if row['group'] == 2:
15         group_sum += row['eggs']
16         group_count += 1
17
18 group_mean = group_sum / group_count
19
20 print(group_mean)
21
22
23
24
25
```

18.95334



```
1  # Pandas udostępnia wiele wbudowanych funkcji, umożliwiających na szybkie obliczenia i inne
2  # operacje na danych.
3  # Funkcja groupby() pozwala nam grupować wartości kolumn na podstawie wartości jednej
4  # z nich.
5  # Funkcja mean() liczy średnią wszystkich elementów danej grupy.
6
7  import pandas
8
9  df = pandas.read_excel('fertilisation.xlsx', 'f_data')
10 # df = pandas.read_csv('fertilisationcsv.csv', sep=';')
11
12 print(df.groupby(['group']).mean())
13
14
15
16
17
18
19
20
21
22
23
24
25
```

|       | water    | eggs      | fertilisation |
|-------|----------|-----------|---------------|
| group |          |           |               |
| 1     | 0.064280 | 12.819453 | 97.0          |
| 2     | 0.109240 | 18.953340 | 95.0          |
| 3     | 0.126353 | 16.715580 | 10.0          |
| 4     | 0.126033 | 6.957633  | 2.0           |

```
1  # Możemy zawęzić działanie funkcji groupby() do jednej kolumny.
2
3
4  import pandas
5
6  df = pandas.read_excel('fertilisation.xlsx', 'f_data')
7  # df = pandas.read_csv('fertilisationcsv.csv', sep=';')
8
9  print(df.groupby(['group'])['eggs'].mean())
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
```

```
group
1    12.819453
2    18.953340
3    16.715580
4     6.957633
Name: eggs, dtype: float64
```

```
1  # Wykorzystując pakiet pandas, odczytaj plik 'coronavirus_cases_13-03-2020.xlsx' (lub
2  # 'coronavirus_cases_13-03-2020.csv'). Oblicz i wypisz na ekranie:
3  # - średni wiek zarażonych obywateli Chin
4  # - średni wiek zarażonych obywateli Chin pochodzących z Wuhan (kolumna R - from Wuhan)
5  # - liczbę zarażonych kobiet z Japonii
6  # - liczbę zarażonych mężczyzn z Japonii
7
8  df = pandas.read_excel('coronavirus_cases_13-03-2020.xlsx', 'Line-list')
9  # df = pandas.read_csv('coronavirus_cases_13-03-2020.csv', sep=';')
10
11  # ...
12
13
14
15
16
17
18
19
20
21
22
23
24
25
```

```
# docs.python.org/3/ - oficjalna dokumentacja
```

Python » English 3.11.1 3.11.1 Documentation »

## Python 3.11.1 documentation

Welcome! This is the official documentation for Python 3.11.1.

**Parts of the documentation:**

- [What's new in Python 3.11?](#)  
*or all "What's new" documents since 2.0*
- [Installing Python Modules](#)  
*installing from the Python Package Index & other sources*
- [Distributing Python Modules](#)  
*publishing modules for installation by others*
- [Extending and Embedding](#)  
*tutorial for C/C++ programmers*
- [Python/C API](#)  
*reference for C/C++ programmers*
- [FAQs](#)  
*frequently asked questions (with answers!)*
- [Library Reference](#)  
*keep this under your pillow*
- [Language Reference](#)  
*describes syntax and language elements*
- [Python Setup and Usage](#)  
*how to use Python on different platforms*
- [Python HOWTOs](#)  
*in-depth documents on specific topics*

**Download**  
Download these documents

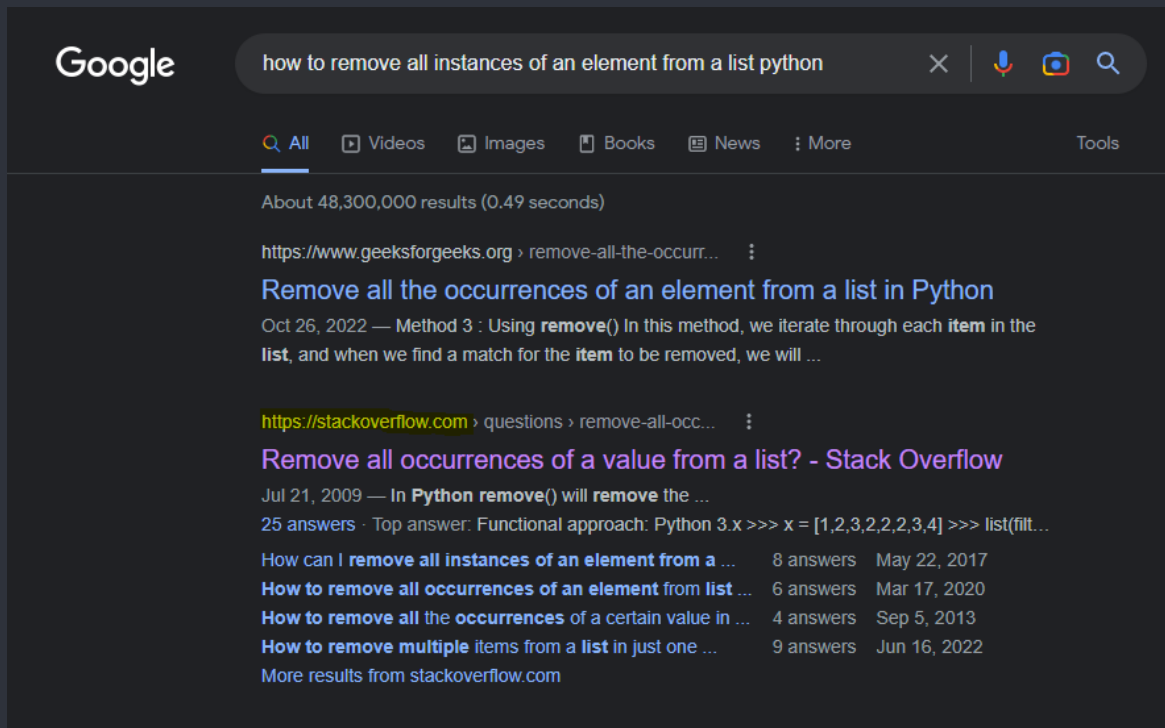
**Docs by version**

- [Python 3.12 \(in development\)](#)
- [Python 3.11 \(stable\)](#)
- [Python 3.10 \(stable\)](#)
- [Python 3.9 \(security-fixes\)](#)
- [Python 3.8 \(security-fixes\)](#)
- [Python 3.7 \(security-fixes\)](#)
- [Python 3.6 \(EOL\)](#)
- [Python 3.5 \(EOL\)](#)
- [Python 2.7 \(EOL\)](#)
- [All versions](#)

**Other resources**

- [PEP Index](#)
- [Beginner's Guide](#)
- [Book List](#)
- [Audio/Visual Talks](#)
- [Python Developer's Guide](#)

```
1 # google.com -> szukamy wątków stackoverflow.com  
2 # - najszybszy sposób na znalezienie rozwiązania  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25
```



## # chat.openai.com – przerażająco inteligentny chat bot



Jak można usunąć element z listy oraz wszystkie jego powtórzenia w Pythonie? Podaj przykładowy kod, bez zastosowania list comprehension.



Aby usunąć element z listy oraz wszystkie jego powtórzenia w Pythonie, można użyć pętli `for` i metody `remove()`. Poniżej znajduje się przykładowy kod:

[Copy code](#)

```
# Przykładowa lista
my_list = [10, 20, 30, 10, 20, 40, 50]

# Element, który chcemy usunąć
element = 20

# Używamy pętli for, aby przejść przez wszystkie elementy listy
for item in my_list:
    # Jeśli dany element jest równy szukanemu elementowi, usuwamy go z listy
    if item == element:
        my_list.remove(item)

# Po zakończeniu pętli, wszystkie wystąpienia elementu zostaną usunięte z listy
print(my_list) # [10, 30, 10, 40, 50]
```

Uwaga: Pamiętaj, że w powyższym kodzie używamy metody `remove()`, która usuwa tylko pierwsze wystąpienie danego elementu. Jeśli element występuje na liście kilka razy, kod będzie go usuwał pojedynczo, aż do momentu, gdy wszystkie wystąpienia zostaną usunięte.

# codewars.com

Kata Training

**Jaden Casing Strings**

2212 327 86% of 17,730 66,514 of 176,577 jacob 16 Issues Reported

Instructions Output

Jaden Smith, the son of Will Smith, is the star of films such as The Karate Kid (2010) and After Earth (2013). Jaden is also known for some of his philosophy that he delivers via Twitter. When writing on Twitter, he is known for almost always capitalizing every word. For simplicity, you'll have to capitalize each word, check out how contractions are expected to be in the example below.

Your task is to convert strings to how they would be written by Jaden Smith. The strings are actual quotes from Jaden Smith, but they are not capitalized in the same way he originally typed them.

Example:

Not Jaden-Cased: "How can mirrors be real if o  
Jaden-Cased: "How Can Mirrors Be Real If O

Solution

```
1 def to_jaden_case(string):  
2     # ...
```

Sample Tests

```
1 from solution import to_jaden_case  
2 import codewars_test as test  
3  
4 @test.describe('Sample test')  
5 def basic_tests():  
6     @test.it('Simple text')  
7     def _():  
8         quote = "How can mirrors be real if our eyes aren't real"  
9         ...
```

Simplify software development with the One DevOps Platform. Start your  
powered by  
Qualified:


```
# rosalind.info
```

The screenshot shows the Rosalind website interface. At the top is a navigation bar with the Rosalind logo, links for 'About', 'Problems', 'Statistics', and 'Glossary', a search bar, and social media icons for Facebook and Twitter. On the right side of the navigation bar are links for 'Log in' and 'Register'.

## Locations


Rosalind is a platform for learning bioinformatics and programming through problem solving. [Take a tour](#) to get the hang of how Rosalind works.

If you don't know anything about programming, you can start at the [Python Village](#). For a collection of exercises to accompany Bioinformatics Algorithms book, go to the [Textbook Track](#). Otherwise you can try to storm the [Bioinformatics Stronghold](#) right now.




### Python Village

If you are completely new to programming, try these initial problems to learn a few basics about the Python programming language. You'll get familiar with the operations needed to start solving bioinformatics challenges in the Stronghold.



### Bioinformatics Stronghold

Discover the algorithms underlying a variety of bioinformatics topics: computational mass spectrometry, alignment, dynamic programming, genome assembly, genome rearrangements, phylogeny, probability, string algorithms and others.




### Bioinformatics Armory

Ready-to-use software tools abound for bioinformatics analysis. Whereas in the Stronghold you implement algorithms on your own, in the Armory you solve similar problems by using existing tools.

### Bioinformatics Textbook Track

A collection of exercises to accompany Bioinformatics Algorithms: An Active-Learning Approach by Phillip Compeau & Pavel Pevzner. A full version of this text is hosted on [stepic.org](#)

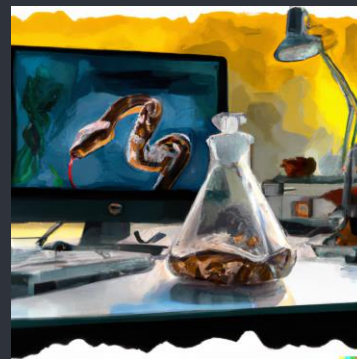
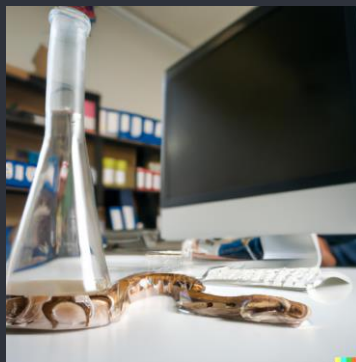
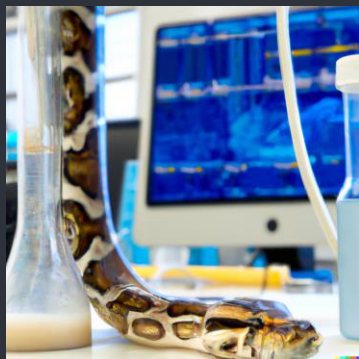


### Algorithmic Heights

A collection of exercises in introductory algorithms to accompany "Algorithms", the popular textbook by Dasgupta, Papadimitriou, and Vazirani.



1 Serdecznie dziękuję! = {



11 „a python crawling out of glass flask in a laboratory, with a computer in the background” by DALL-E 2

12 e-mail : ['mateusz.dobrychlop@gmail.com'],

13 linkedin : ['linkedin.com/in/mdobrychlop'],

14 }

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**