PyStart #47 Więcej o CSV

Lepsze odczytywanie plików CSV

- Pliki CSV często mają wyrażenia ujęte w “” oznacza to to, że nawet jeśli separator pojawi się w tym tekście to nie powinien być on uznany za *rozdzielający*.
- W ten sposób możemy również zapisywać dane w pliku CSV znacznie wygodniej.
- Jeśli plik posiada wiersz nagłówkowy to dzięki temu możemy też odczytywać plik jako słownik.

PyStart #47 Więcej o CSV

Odczyt pliku CSV

Delimiter oraz quotechar pozwalają nam na ustawienie separatora oraz znaku który sprawi, że Python zinterpretuje wartość jako tekst.

```
1   import csv
2
3   with open('orders.csv', newline='') as csvfile:
4       reader = csv.reader(csvfile, delimiter=';', quotechar='"')
5       for row in reader:
6           print(row)
```

PyStart #47 Więcej o CSV

Zapis do CSV

Analogicznie możemy też w bardzo wygodny sposób dopisywać nowe wiersze jakie powinny się znaleźć w naszym pliku.

```
1  import csv
2
3  with open('orders.csv', mode='w', newline='') as csvfile:
4      writer = csv.writer(csvfile, delimiter=';', quotechar='"')
5      writer.writerow([4, 'Kacper', 'Sieradziński'])
6
```

PyStart #47 Więcej o CSV

Odczyt pliku CSV

Jeśli pierwszy wiersz jest nagłówkowym to możemy również odczytywać poszczególne **dalsze** wiersze jako słowniki.

```
1   import csv
2
3   with open('orders.csv', mod newline='') as csvfile:
4       reader = csv.DictReader(csvfile, delimiter=';', quotechar='"')
5       for row in reader:
6           print(row['first_name'], row['last_name'])
7
```

PyStart #47 Więcej o CSV

Zapis pliku CSV

Podobnie można również zapisywać nowe wiersze jako słowniki.

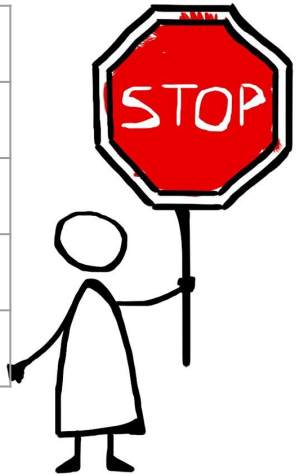
```python
import csv

with open('orders.csv', mode='w', newline='') as csvfile:
    fieldnames = ['first_name', 'last_name']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writerow({'first_name': 'Kacper', 'last_name': 'Sieradziński'})
```

PyStart #47 Więcej o CSV

Zadania dla nabrania wprawy

Wykorzystaj zdobytą wiedzę o plikach CSV aby utworzyć nowy plik zawierający poniższe dane, a następnie w drugim programie odczytaj wszystkie zgromadzone informacje.

Name	time_to_100	speed_record
Hennessey Venom F5	2.6	484
SSC Tuatara	2.5	460
Koenigsegg Agera RS	3,1	457
Koenigsegg One 1	2,6	450



PyStart #48 Wykonywanie poleceń systemowych

Dlaczego?

- ➔ Automatyzacja, np. wykonywanie polecenia o określonym czasie.
- ➔ Integrowanie gotowych narzędzi, które nie udostępniają nam API.
- ➔ Uruchamianie i zatrzymywanie programów.
- ➔ Kontrolowanie procesów uruchomionych przez Pythona.
- ➔ Wykonywanie takich poleceń jak np. `ping`

PyStart #48 Wykonywanie poleceń systemowych

Przykładowe zastosowanie

- Skrypt wykonujący backup danych - kopiowanie danych w konkretne miejsce, a następnie pakowanie danych za pomocą polecenia tar.gz i usuwanie rozpakowanych danych.
- Automatyczne pobieranie plików z konkretnego adresu za pomocą polecenia wget.
- Możesz np. sprawdzać listę aktualnie włączonych procesów za pomocą linuksowego polecenie `ps` lub windowsowego `tasklist`

PyStart #48 Wykonywanie poleceń systemowych

Jak to robić?

```
import subprocess

# Nie otrzymamy wyniku zapytania, wyświetli się ona na ekranie
# Nie można będzie przetworzyć wyniku.
subprocess.run(‘ls’, shell=True)
subprocess.run([‘ls’, ‘-l’], shell=True)


# Zmienna result będzie przechowywała wynik, zwróć uwagę na capture_output
result = subprocess.run(["ls", "-l], shell=True, capture_output=True)
```

Uważaj na wykonywane polecenia. Jeśli użytkownik ma wpływ na to co zostanie wykonane może w ten sposób zaatakować nasz system.

PyStart #48 Wykonywanie poleceń systemowych

Odbieranie odpowiedzi jako tekst

```
# Zmienna result będzie przechowywała wynik, zwróć uwagę na capture_output
result = subprocess.run(["ls", "-l], shell=True, capture_output=True)
print(result.stdout.decode(‘utf8’))
```

PyStart #48 Wykonywanie poleceń systemowych

Zadania dla nabrania wprawy

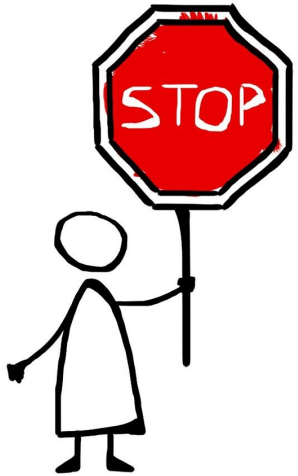
Odbierz od użytkownika adres IP lub domenę np. `pystart.pl`, a następnie wykonaj polecenie systemowe `ping <podany adres>`

np. `ping pystart.pl`

Wyświetl informację “Adres jest dostępny” lub “Adres jest niedostępny” w zależności od tego czy adres odpowiedział lub nieodpowiedział.

```
C:\Users\Kacper>ping pystart.pl

Pinging pystart.pl [109.95.156.170] with 32 bytes of data:
Reply from 109.95.156.170: bytes=32 time=27ms TTL=59
Reply from 109.95.156.170: bytes=32 time=25ms TTL=59
Reply from 109.95.156.170: bytes=32 time=24ms TTL=59
```

```
C:\Users\Kacper>ping pystart_no.pl
Ping request could not find host pystart_no.pl. Please check the name and try again.
```



PyStart #49 Argparse

Kiedy się przydaje?

- Programy, które uruchamiamy mogą otrzymywać argumenty uruchomieniowe.

```
python my_program.py --name=Kacper --age 18
```

- W ten sposób tworzymy tzw. narzędzia cli, programy którymi możemy sterować z linii poleceń w konsoli.

- Argparse dostarcza nam również pomoc (`python my_program.py -h`) wyświetlającą arkusz zawierający podpowiedzi

PyStart #49 Argparse

Przykład?

```python
import argparse

parser = argparse.ArgumentParser(
    prog='Hello World!',
    description='This is an sample Pystart application!',
    epilog='Author: Kacper Sieradziński'
)

parser.add_argument('name')
parser.add_argument('-l', '--lastname')
parser.add_argument('-a', '--admin', action='store_true')

args = parser.parse_args()
print(args)
```

```python
parser.add_argument('-a', '--year', type=int)
```

- ➔ `name` to argument wymagany
- ➔ `lastname` i `admin` to argumenty opcjonalne

PyStart #49 Argparse

Jak to odebrać?

```
import argparse

parser.add_argument('name')
parser.add_argument('-l', '--lastname')
parser.add_argument('-a', '--admin', action='store_true')

args = parser.parse_args()
print(args.name)
print(args.lastname)


args = parser.parse_args()
print(args)

```

```
parser.add_argument('-a', '--year', type=int)
```

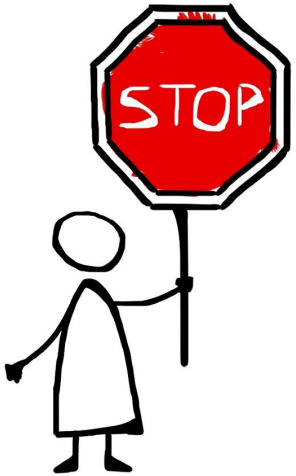
PyStart #49 Argparse

Zadania dla nabrania wprawy

Przygotuj program, który będzie odbierał argumenty takie jak:

- `path` (wymagany) (string)
- `target` (wymagany) (string)
- `width` (integer) (domyślnie 300)
- `height` (integer) (domyślnie 300)

Na chwilę obecną wyłącznie wyświetl te wartości jedna pod drugą



PyStart #50 Praca z filesystemem

Kiedy się przydaje?

- ➔ Chcesz wylistować pliki w katalogu?
- ➔ Potrzebujesz segregować pliki w zależności od rozszerzeń?
- ➔ Szukasz pliku zawierającego konkretną nazwę?
- ➔ Nie warto parsować polecenia `dir` czy też `ls`, dużo wygodniej użyć `os.walk`

PyStart #50 Praca z filesystemem

Listowanie zawartości

```
from os import walk

for dir, directories_in_dir, files_in_dir in walk('./some_path'):
    ....
```

Walk przechodzi po ścieżkach rekursywnie, tzn. wchodzi do wszystkich katalogów wewnątrz `some_path`

- ➔ `dir` - ścieżka do przetwarzanego folderu
- ➔ `directories_in_dir` - lista folderów w `dir`
- ➔ `files_in_dir` - lista plików w `dir`

PyStart #50 Praca z filesystemem

Jak sprawdzić rozszerzenie pliku?

- ➔ Nazwa pliku to `string`, można więc wykonać `filename.endswith('.pdf')`
- ➔ Można zrobić to ciut profesjonalniej używając `pathlib` (więcej możliwości)

```python
from pathlib import Path

file = Path('test.txt')

# rozszerzenie pliku
print(file.suffix)

# czy plik istnieje?
print(file.exists())

# ścieżka bezwzględna do pliku
print(file.resolve())

# true lub false czy jest to folder czy plik
print(file.is_file())
print(file.is_dir())
```

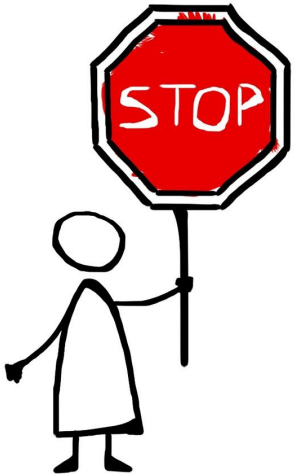
PyStart #50 Praca z filesystemem

Listowanie zawartości

Napisz program, który odbierze od użytkownika ścieżkę do konkretnego katalogu oraz rozszerzenie pliku.

np. `python find_files.py --dir=sample --extension=jpg`

- ➔ Wynikiem operacji powinno być wyświetlenie wszystkich plików posiadających takie rozszerzenie.
- ➔ Odpowiednie dane możesz spreparować osobiście lub użyć katalogu, który już masz gdzieś na dysku.



PyStart #51 Praca z obrazem

Skalowanie obrazu

```
pip install Pillow
```

https://pillow.readthedocs.io/en/stable/

PyStart #51 Praca z obrazem

Skalowanie obrazu

```
1  from PIL import Image
2
3  im = Image.open('flower.jpg')
4  thumbnail = im.resize((300, 300))
5  thumbnail.save('resized.jpg')
6
```

PyStart #51 Praca z obrazem

Zadania dla nabrania wprawy

Przygotuj folder o nazwie “`to_scale`” i umieść w nim dowolne 5-6 plików ze zdjęciami. Ważne by zdjęcia miały rozmiar większy niż 800 pikseli szerokości.

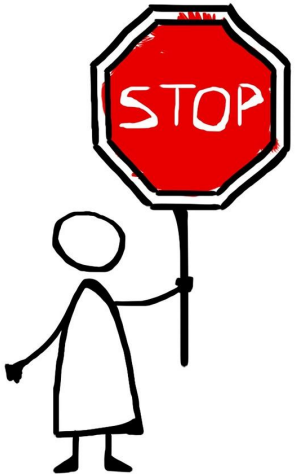
Korzystając z blibliotek: `Pillow` oraz `os.walk` przejdź po wszystkich plikach w tym katalogu, a następnie zapisz miniaturki w folderze thumbnails. Pliki powinny mieć te same nazwy.

`to_scale`

- `image.png`
- `company.png`
- `something.png`

`thumbnails`

- `image.png`
- `company.png`
- `something.png`



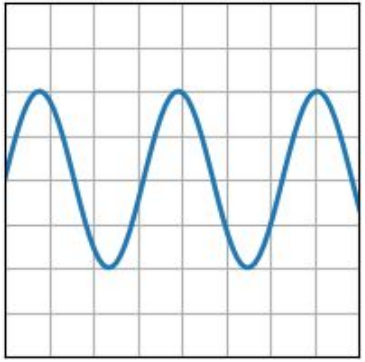
PyStart #52 Tworzenie wykresów

Wykresy w Pythonie

```
pip install matplotlib
```

PyStart #52 Tworzenie wykresów

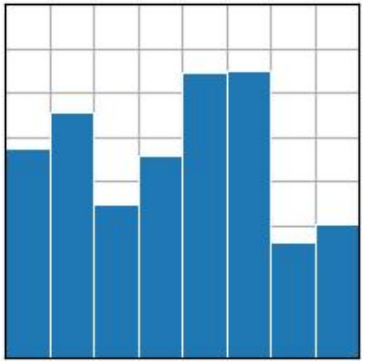
Przykłady



```python
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('_mpl-gallery')

# make data
x = np.linspace(0, 10, 100)
y = 4 + 2 * np.sin(2 * x)

# plot
fig, ax = plt.subplots()

ax.plot(x, y, linewidth=2.0)

ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 8), yticks=np.arange(1, 8))

plt.show()
```

PyStart #52 Tworzenie wykresów

Przykłady



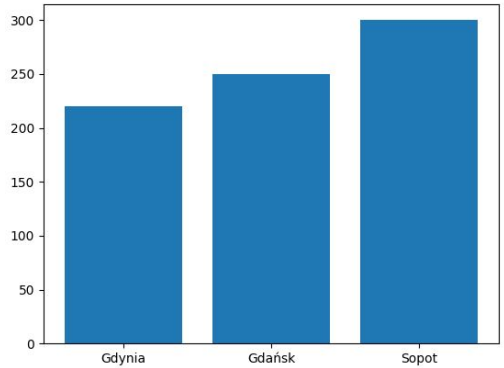
```python
import matplotlib.pyplot as plt
import numpy as np
plt.style.use('_mpl-gallery')

# make data:
np.random.seed(3)
x = 0.5 + np.arange(8)
y = np.random.uniform(2, 7, len(x))

# plot
fig, ax = plt.subplots()

ax.bar(x, y, width=1, edgecolor="white", linewidth=0.7)

ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 8), yticks=np.arange(1, 8))

plt.show()
```

PyStart #52 Tworzenie wykresów

Zróbmy wspólnie coś prostrzego

Wykres zawierający trzy miasta wraz z wartościami.

```python
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

cities = ['Gdynia', 'Gdańsk', 'Sopot']
people = [220, 250, 300]

ax.bar(cities, people)
plt.show()
```

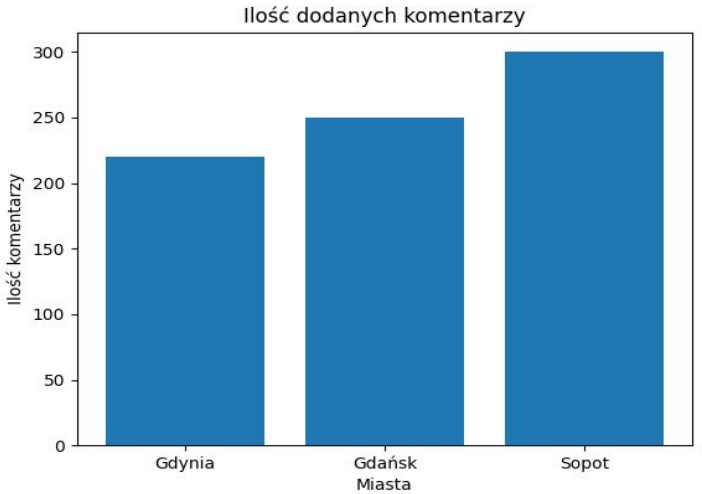


PyStart #52 Tworzenie wykresów

Zróbmy wspólnie coś prostrzego

Dodajmy do wykresu informacje

```python
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

cities = ['Gdynia', 'Gdańsk', 'Sopot']
people = [220, 250, 300]

ax.set_xlabel('Miasta')
ax.set(ylabel='Ilość komentarzy', title='Ilość dodanych komentarzy')
ax.bar(cities, people)
plt.show()
```



PyStart #52 Tworzenie wykresów

Zapisywanie wykresu do pliku

Aby zapisać wykres do pliku zamiast lub poza instrukcją:

```
plt.show()
```

warto dodać drugą:

```
plt.save_fig('nazwawykresu.png')
```

Gdzie `nazwawykresu.png` to oczywiście nazwa pliku,

którą dostanie nasz wykres.

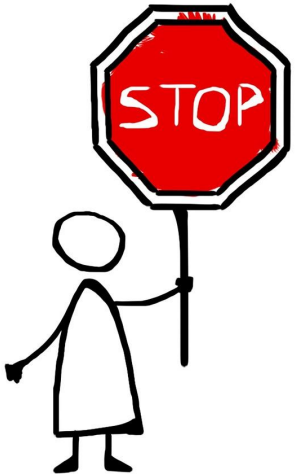
PyStart #52 Tworzenie wykresów

Zadania dla nabrania wprawy

- Otwórz dołączony do zadania plik `operacje.csv`. Policz wartość dochodu dla poszczególnych miesięcy, a następnie utwórz wykres prezentujący dochód dla tych miesięcy.

- Utwórz drugi wykres ilustrujący ilość operacji wykonywanych w poszczególnych miesiącach.

- Zapisz oba wykresy w katalogu pod nazwami “`podsumowanie.png`” oraz “`ilosc_operacji.png`”



PyStart #53 Budowa API

Jak stworzyć własną usługę?

```
pip install Flask
```

PyStart #53 Budowa API

Przetwarzanie daty i czasu inaczej

```python
import pendulum

dt = pendulum.now()
print(dt.format('dddd DD MMMM YYYY', locale='pl'))

```

PyStart #53 Budowa API

Przetwarzanie daty i czasu inaczej

```python
import pendulum

dt = pendulum.now()
print(dt.format('dddd DD MMMM YYYY', locale='pl'))

```

PyStart #54 Bazy danych - sqlite

Czym jest sqlite?

- ➔ Baza danych relacyjna.
- ➔ Dane przechowywane mogą być w pliku lub w pamięci.
- ➔ Do odczytania danych potrzebujemy programu, ja używam DBeaver’a.
- ➔ Do pobierania / zapisywania danych używamy języka zapytań jakim jest SQL.
- ➔ Obsługa sqlite w Pythonie, stanowi część biblioteki standardowej.

PyStart #54 Bazy danych - sqlite

Odrobina terminologii

Baza danych - 1 plik (np. pystart)

tabela - users
kolumny:
- id
- login
- password
- first_name
- last_name
- email

tabela - courses
kolumny:
- id
- title
- category
- start_at
- end_at
-

reports

vouchers

posts

files

lessons

itd..

PyStart #54 Bazy danych - sqlite

Co wykonamy?

➔ Stworzymy aplikację, która będzie przechowywała stan naszej biblioteczki w bazie danych:

- Podłączymy się do bazy danych (Utworzymy nową bazę)
- Utworzymy potrzebną tabelę
- Wykonamy polecenie dodające konkretną książkę do bazy
- Wykonamy polecenie listujące książki znajdujące się w naszej tabeli
- Więcej? Bootcamp: SkumajBazy

PyStart #54 Bazy danych - sqlite

Podłączenie do bazy danych

```python
import sqlite3

connection = sqlite3.connect('library.db')
connection.close()
```

lub

```python
import sqlite3

with sqlite3.connect('library.db') as connection:
    ...
```

PyStart #54 Bazy danych - sqlite

Odrobina SQL - utworzenie bazy

```sql
CREATE TABLE books(
    book_id INTEGER PRIMARY KEY AUTOINCREMENT,
    title VARCHAR(100) UNIQUE NOT NULL,
    author VARCHAR(100)
)
```

- ➔ PRIMARY KEY - coś co wyróżnia nasz rekord
- ➔ UNIQUE - jeśli dodam drugą książkę o tym samym tytule dostanę błąd
- ➔ NOT NULL - wartość jest wymagana

PyStart #54 Bazy danych - sqlite

Odrobina SQL - dodajemy książkę

```sql
INSERT INTO
    books(title, author)
VALUES
    ('W pustyni i w puszczy', 'Henryk Sienkiewicz')
```

lub

```sql
INSERT INTO
    books
VALUES
    (NULL, 'W pustyni i w puszczy', 'Henryk Sienkiewicz')
```

PyStart #54 Bazy danych - sqlite

Odrobina SQL - pobieramy książki

```sql
SELECT
    book_id, title, author
FROM books
```

pobranie z uwzględnieniem warunku

```sql
SELECT
    book_id, title, author
FROM books
WHERE book_id = 1
```

PyStart #54 Bazy danych - sqlite

Wywoływanie zapytań w Pythonie

```python
import sqlite3

with sqlite3.connect('library.db') as connection:
    cursor = connection.cursor()
    books = cursor.execute('SELECT book_id, title_author FROM books')

    for book in books:
        print(book)

```

PyStart #54 Bazy danych - sqlite

Wywoływanie zapytań w Pythonie

```python
import sqlite3

with sqlite3.connect('library.db') as connection:
    cursor = connection.cursor()
    title = input('Tytuł: ')
    author = input('Autor: ')
    cursor.execute('INSERT INTO books VALUES (null, ?, ?)', (title, author))
    connection.commit()

```

commit!

ÓSMA PRACA DOMOWA

UWAGA! UWAGA!

- Przygotuj grę wisielec z interfejsem graficznym. Gra powinna losować jedno z haseł zapisywane w kodzie. Jeśli użytkownik trafi literę znajdującą się w haśle, litera ta powinna być odkrywana. W przeciwnym wypadku zdobywa kolejną z liter W - I- S-I-E-L-E-C. Gra kończy się gdy użytkownik odgadnie hasło nim skompletuje całego “WISIELCA”.
- Pytaj użytkownika o nazwy produktów tak długo aż nie napisze “koniec”. Zapisz wówczas do pliku o nazwie według wzoru ddmmrrrr.txt tylko unikatowe nazwy wprowadzonych przez niego produktów.
- Znajdź sposób listowania plików w katalogu, a następnie napisz program, który odszuka wszystkie pliki tekstowe w wybranym przez Ciebie folderze i do nowego pliku o nazwie scalone.txt wstawi wynik jakim będą połączone te pliki.
- Zmodyfikuj grę wisielec aby hasła były losowane z pliku tekstowego.
- Przygotuj program, który przeskanuje katalog pod kątem różnych rozszerzeń. Wynikiem powinien być wykres prezentujący te rozszerzenia.