Pliki w Pythonie

open, os.path i pathlib, czyli z której strony ugryźć dostęp dysku

open, czyli magia dostępu do dysku

- open -> wbudowana funkcja, służąca do dostępu do plików

```
def open(file, mode='r', encoding=None, ...)
```

- Co zwraca open?

```
>>> open("test.txt")
<_io.TextIOWrapper name='test.txt' mode='r'
encoding='cp1250'>
```

Co zostało napisane?

- Open z „r”

```
file = open("test.txt", "r")

file.read(5)
file.read()

line = file.readline()
while line != "":
    print(line)
    line = file.readline()
```

Co zostało napisane?

- Czytanie wszystkich linii

```
line = file.readline()
while line != "":
    print(line)
    line = file.readline()

for line in file.readlines():
    print(line)

for line in file:
    print(line)
```

Gdzie mam to wrzucić?

- Open z „w”

```
file = open("test.txt", "w")

file.write("mój ulubiony tekst")
file.writelines(["każda", "linijka", "jest", "ważna"])
```

Gdzie mam to wrzucić?

- Open z „w” a open z „a”

```
file = open("test.txt", "w")
# zapisujemy "kot" w pliku
file = open("test.txt", "r")
print(file.read())

file = open("test.txt", "a")
# zapisujemy "kot" w pliku
file = open("test.txt", "r")
print(file.read())
```

Czy zmiany zostały zapisane?

- Czyli czemu ważne jest file.close()

```
file = open("test.txt", "w")

file.write("mój ulubiony tekst")

file.close()
```

Nasze ukochane błędy

- Czytanie plików do których nie mam dostępu: PermissionError
- Czytanie nieistniejących plików: FileNotFoundError
- Czytanie plików binarnych lub złe kodowanie: UnicodeDecodeError

Praca z plikami dobre praktyki

- Pamiętajmy o zamykaniu otwartych plików!

```
file = open("plik.txt", "r")
# operacje na pliku
file.close()
```

- Jeśli mamy długo działający program, który trzyma otwarty plik .flush() pomoże upewnić się, że nie zgubimy danych

```
for idx, line in enumerate(data):
    file.write(line)
    if (idx + 1) % 10 == 0:
        file.flush()
```

`with ... as ...:` czyli nie chcę pamiętać o zamykaniu plików

```
file = open("plik.txt", "r")
for line in file:
    print(line)
file.close()
```

kontra

```
with open("plik.txt", "r") as file:
    for line in file:
        print(line)
```

Przypomnienie o ścieżkach

- Ścieżki względne:

```
./ala/ma/kota
ala/ma/kota
```

- Ścieżki bezwzględne:

```
/tmp/ala/ma/kota
C:\ala\ma\kota
```

os i os.path, czyli czas wędrować po dysku

- os.path.join() czyli nie chcę zastanawiać się na jakim systemie

```
# UNIX
os.path.join("ala", "ma", "kota") -> "ala/ma/kota"
# WINDOWS
os.path.join("ala", "ma", "kota") -> "ala\\ma\\kota"
```

- os.getcwd() czyli gdzie jestem?

```
os.getcwd() -> "/users/test"
```

- os.listdir() co widzę dookoła mnie?

```
os.listdir(path) -> ["Downloads", "test.py", ...]
```

os i os.path, czyli czas wędrować po dysku

- os.chdir(directory) czyli jak zmienić katalog?

```
os.chdir("ala/ma/kota")
```

- os.path.isdir(path) i podobne, czyli czy co leży pod ścieżką?

```
os.path.isdir("ala/ma/kota") -> True
os.path.isfile("ala/ma/kota") -> False
```

Ścieżki jako obiekty, czyli pathlib

- Zaczynamy:

```
from pathlib import Path
```

- Dzielenie sposobem na składanie ścieżki

```
path = Path("ala") / "ma" / "kota"
path = Path("ala/ma/kota")
path = Path("ala") / Path("ma/kota")
# ALE UWAGA!!!
path = Path("ala") / Path("/ma/kota")
```

Ścieżki jako obiekty, czyli pathlib

- Path.open() czyli open(), ale wygodniejszy

```
path.open("r")
# Kontra
open("ala/ma/kota.txt", "r")
```

- Path.is_dir() i is_file(), czyli co trzymam w ręku?

```
path.is_file()
path.is_dir()
# Kontra
os.path.isdir("path")
os.path.isfile("path")
```

Ścieżki jako obiekty, czyli pathlib

- Path.write_text(), Path.read_text() czyli jeszcze mniej zastanawiania się nad tym co zrobić z otwartym plikiem

```
path.write_text("mój ulubiony tekst")

text = path.read_text()
```

Q&A?