Python_Codzienny()

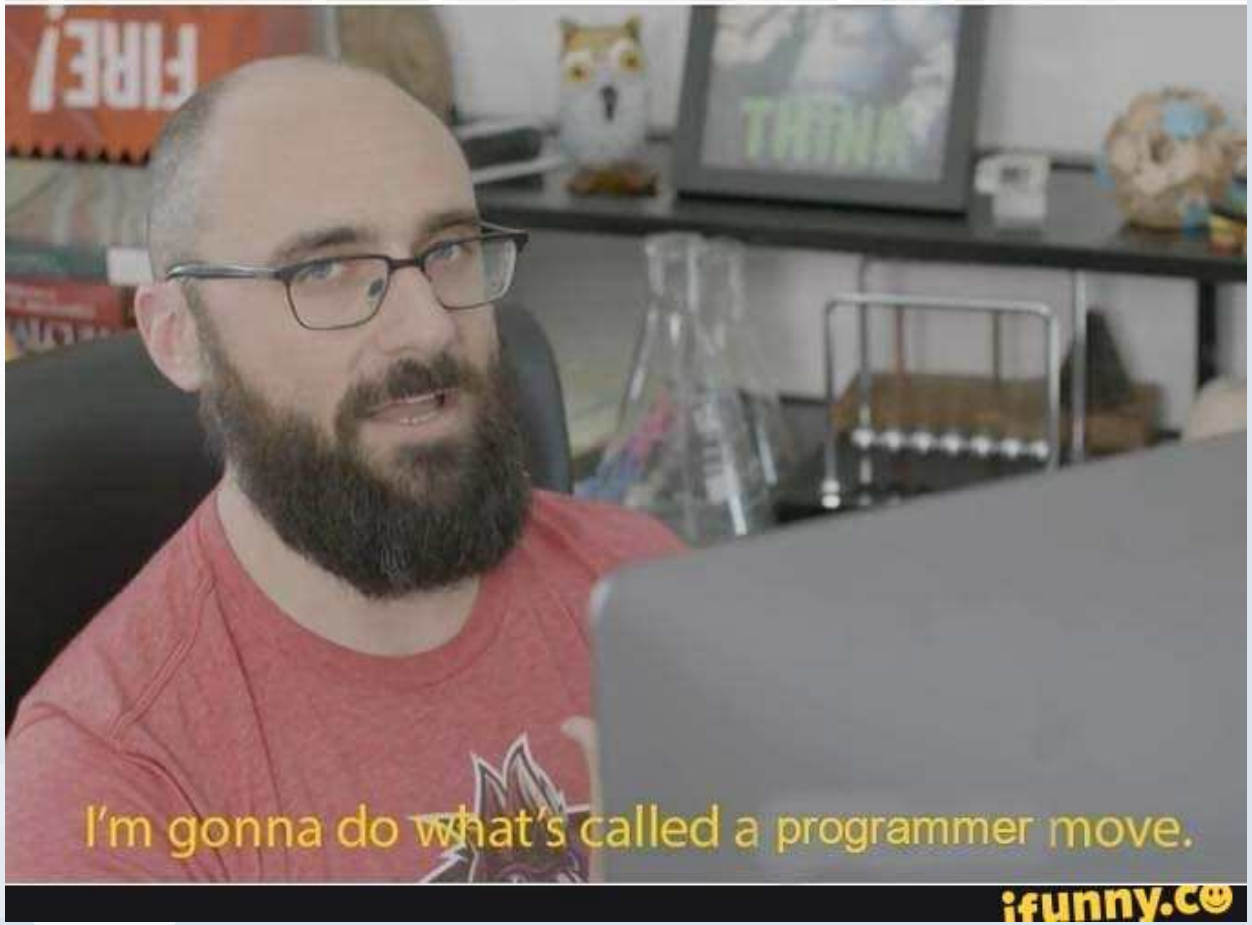
- Mikołaj Storoniak

Nudne, powtarzalne zadania

Nudne, powtarzalne zadania

- Zmiana nazw plików
- Sortowanie/przerzucanie plików
- Kopiowanie danych z internetu
- Zmiana formatowania tekstu
- Tworzenie wykresów na podstawie danych
- Ładowanie plików do chmury

When there's a task that can be done manually in 10 minutes but you find a way to automate it in 10 days



Praca z plikami

- Przykładowe zastosowania:
  - Tworzenie zaproszeń
  - Wysyłanie wielu podobnych maili
  - Dzielenie dużego pliku na mniejsze
  - Konfiguracja aplikacji i zapisywanie jej danych
  - Zmiana formatu pliku

Otwieranie pliku

- Aby otworzyć plik używamy `open(file, mode)`
  - File (string): ścieżka do pliku
  - Mode (string): tryb, w którym otwieramy plik. Mode może składać się z wielu znaków

Character	Meaning
`'r'`	open for reading (default)
`'w'`	open for writing, truncating the file first
`'x'`	open for exclusive creation, failing if the file already exists
`'a'`	open for writing, appending to the end of file if it exists
`'b'`	binary mode
`'t'`	text mode (default)
`'+'`	open for updating (reading and writing)

Otwieranie pliku

- Opcje otwierania pliku:
  - ‘r’: Read. Dane są czytane od początku pliku. ⬆
  - ‘w’: Write. Zawartość pliku zostaje usunięta i istnieje możliwość nadpisania jej nową. Jeśli plik nie istnieje, zostaje stworzony. ⬇
  - ‘a’: Append. Druga opcja zapisu, jednak dane zostają dopisane na końcu pliku (plik nie jest czyszczony).
  - ‘+’: Dodaje możliwość zapisu/odczytu. Położenie kursora zależne od wybranego trybu.
  - ‘t’: dane będą traktowane jak tekst (domyślne)
  - ‘b’: dane będą traktowane jak ciąg bitów (np. Jeśli otwieramy obrazek)

Odczyt / Zapis

- Funkcja `open()` zwraca obiekt reprezentujący plik
- Na tym obiekcie możemy przeprowadzać dalsze operacje, w zależności od wybranego trybu:
  - `read(x)` odczytuje x znaków z pliku. Jeśli nie podamy x, odczytuje całość.
  - `readline()` odczytuje kolejną linię.
  - `readlines()` zwraca listę linijek z pliku
  - `write(x)` zapisuje x do pliku
- Po zakończeniu pracy należy zamknąć plik przy pomocy metody `close()`

Odczyt / Zapis

- Istnieje wiele innych metod obiektu pliku (ale od tego jest dokumentacja)

Method	Description
close()	Closes the file
detach()	Returns the separated raw stream from the buffer
fileno()	Returns a number that represents the stream, from the operating system's perspective
flush()	Flushes the internal buffer
isatty()	Returns whether the file stream is interactive or not
read()	Returns the file content
readable()	Returns whether the file stream can be read or not
readline()	Returns one line from the file
readlines()	Returns a list of lines from the file
seek()	Change the file position
seekable()	Returns whether the file allows us to change the file position
tell()	Returns the current file position
truncate()	Resizes the file to a specified size
writable()	Returns whether the file can be written to or not
write()	Writes the specified string to the file
writelines()	Writes a list of strings to the file

Pliki - zadanie

- Stwórz plik przy pomocy open()
- Zapisz do niego kilka linijek tekstu (jakichkolwiek).
- Znowu otwórz plik
- Odczytaj zawartość pliku i wyświetl ją na ekranie
- Dopisz kilka linijek do pliku i powtórz poprzedni krok.

- Bonus: spróbuj zrobić to, nie zamykając pliku
  - `file.seek(A,B)` - przesuwa kursor o A pozycji od punktu B
    B: 0 = początek pliku, 1 = aktualna pozycja, 2 = koniec pliku

`read_write.py`

Operacje na tekście

Operacje na tekście

- Poniższe funkcje stanowią metody klasy `str`
- `capitalize()/lower()/upper()` – zmiana wielkości liter
- `strip()` – usuwa białe znaki
- `split(x)` – zmienia str w tablicę, dzieląc go na znaku x
- `join(x)` – łączy tablicę x w jednego str
- `replace(x,y)` – zamienia wszystkie wystąpienia x na y
- `index(x) / find(x)` – zwraca pozycję na której jest x (index działa też na listach!)
- `format(x,y,z...)` – podmienia oznaczone miejsca w tekście na kolejne argumenty x,y,z...
  - Istnieje wiele opcji formatowania, warto zapoznać się z dokumentacją

`text_operations.py`

Operacje na tekście – zadnie

- Na GitHubie znajdziesz plik – dates.txt
- Zapisano w nim imiona 100 osób oraz daty ich urodzenia
- Lata, miesiące, dni oraz imiona, oddzielone średnikiem, zapisano w osobnych linijkach.
- Dane są uszkodzone – zamiast zer pojawiają się literki „o”, a wielkość znaków w imionach jest losowa
- Przetwórz i zapisz do osobnego pliku dane tak, żeby w każdej linijce było jedno imię i jedna data
- Format powinien wyglądać tak: imię: rok-miesiąc-dzień (bonusowe punkty za użycie polecenia format)

Wykresy

- Często zdarza się, że podczas obrabiania danych musimy je zwizualizować
- Z pomocą przychodzi nam biblioteka matplotlib
- Zwykle importujemy ją jako
  `import matplotlib.pyplot as plt`
- `Matplotlib` to biblioteka kodu, zaś `pyplot` stanowi interfejs, który zachowuje stan między wywołaniami

Podstawowe funkcje

- `plt.plot(x,y)` – tworzy wykres liniowy, x i y to tablice wartości
- `plt.show()` – pokazuje utworzony wcześniej wykres i zatrzymuje program aż do jego zamknięcia
- `plt.hist(x, n_bins)` – tworzy histogram
- `plt.pie(x, labels)` – tworzy wykres kołowy
- `plt.title(str)` – nadaje tytuł wykresowi
- `plt.xlabel(str), plt.ylabel(str)` – etykiety osi
- `plt.axis(arr)` – wartośći na osiach. `arr` to tablica: `[xmin,xmax,ymin,ymax]`

Pyplot

- W tym przykładzie korzystamy z interfejsu Pyplot.
- Istnieją inne, np. Pylab
- Można używać „samego” Matplotlib, ale Pyplot ułatwia nam życie
- Sam Matplotlib może się przydać, jeśli wykres ma stanowić element większej aplikacji

Pyplot

- Pyplot zachowuje swój stan między wywołaniami
- Mamy jeden wykres „w pamięci”, który modyfikujemy każdą kolejną wywołaną funkcją
- `plt.show` wyświetla zapamiętany wykres
- Utworzenie nowego wykresu sprawi, że Pyplot zapomni o poprzednim

Zadania – Wykres Sinusa

- Na Githubie znajdziesz plik „sinus_data.csv”, który zawiera współrzędne punktów wykresu funkcji sinus.
- Pierwszy wiersz zawiera wartości osi x, zaś drugi wartości osi y. Liczby są oddzielone średnikami.
- Należy narysować wykres tej funkcji
- Bonusowe punkty za dodanie etykiet osi i wyświetlenie funkcji tylko dla $x \in \langle \pi, 2\pi \rangle$

text_operations.py

Zadanie - Litery

- Plik „abc_data.csv” zawiera ciąg liter ‘a’, ‘b’ i ‘c’ oddzielonych średnikami.
- Należy stworzyć historgram przedstawiający liczbę wystąpień każdej litery
- Następnie należy stworzyć analogiczny wykres kołowy opatrzony odpowiednimi etykietami.
- Uwaga – histogram jest „sprytny” i wystarczy podać mu tablicę liter + liczbę „koszyków”. Dla wykresu kołowego trzeba je najpierw policzyć.

letters.py

Zadanie – rozkład standardowy

- Należy stworzyć kolejny histogram, tym razem przedstawiający rozkład standardowy
- Można go stworzyć przy pomocy `np.random.standard_normal(10000)` z biblioteki numpy
- Sprawdź co się stanie, kiedy zmienisz liczbę „kubełków”

Źródła

- https://zapier.com/blog/python-automation/ (!!!!!!!!!)
- https://docs.python.org/3/
- https://stackoverflow.com/questions/1466000/difference-between-modes-a-a-w-w-and-r-in-built-in-open-function
- www.geeksforgeeks.org
- https://www.w3schools.com/
- https://shotkit.com/free-raw-photos/
- https://stackoverflow.com/questions/2547349/what-does-x-mean-in-c-c