Zestaw 3

1. W pliku tramwaje.json znajdują się dane z numerami aktualnie kursujących linii tramwajowych w Krakowie oraz przystanków, przez które przejeżdża dany tramwaj (stan na 5.11.2023). W języku Python, czytanie danych w formatach .json czy .csv jest wykonywane z pomocą modułów. W naszym przypadku:

```
import json
with open('tramwaje.json', "r", encoding='utf-8') as read_file:
        data = json.load(read_file)
```

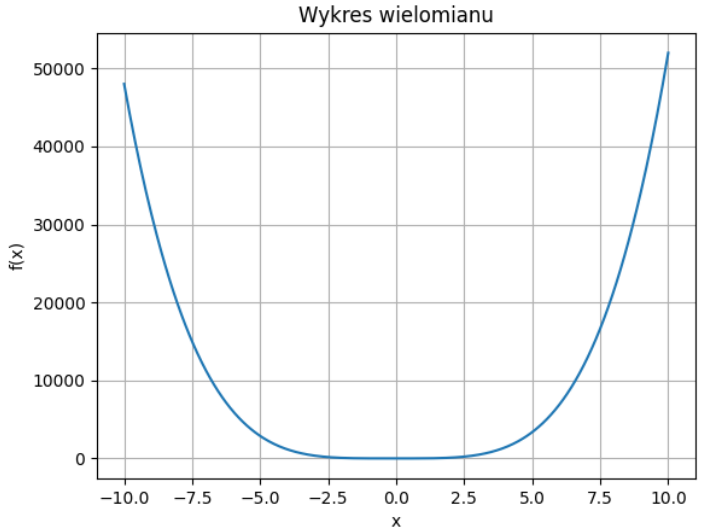
Wczytane dane (zróbmy print) są złożone z zagnieżdżonych typów: słownika, listy, słownika, listy: `{'tramwaje': [{'linia': '1', 'przystanek': [{'nazwa': 'Wańkowicza 01'}, {'nazwa': 'Cienista 01'}, {'nazwa': 'Teatr Ludowy 01'}, …`
Zatem, przykładowo, żeby odczytać pierwszy przystanek dla linii 1, trzeba wywołać w konsoli:
`data['tramwaje'][0]['przystanek'][0]['nazwa']` żeby zobaczyć nazwę `'Wańkowicza 01'`.
Należy przepisać dane do uproszczonego formatu typu słownik, którego kluczem będzie numer linii tramwajowej (zapisany jako int), a wartością krotka zawierająca wszystkie nazwy przystanków danej linii. Uwaga: technicznie przystanki oprócz nazw mają też numery, proszę uprościć dane, zapisując wyłącznie nazwy przystanków, **bez** końcowych numerów (01, 02…). Przykładowo, dla linii nr 1 spodziewany format danych wygląda następująco: `{1: ('Wańkowicza', 'Cienista', 'Teatr Ludowy', 'Rondo Kocmyrzowskie im. Ks. Gorzelanego', 'Bieńczycka', 'Rondo Czyżyńskie', 'Centralna', 'Rondo 308. Dywizjonu', 'M1 al. Pokoju', 'TAURON Arena Kraków al. Pokoju', 'Dąbie', 'Ofiar Dąbia', 'Fabryczna', 'Francesco Nullo', 'Teatr Variété', 'Rondo Grzegórzeckie', 'Hala Targowa', 'Starowiślna', 'Poczta Główna', 'Plac Wszystkich Świętych', 'Filharmonia', 'UJ / AST', 'Muzeum Narodowe', 'Oleandry', 'Park Jordana', 'Reymana')}`
Proszę wynik konwersji zapisać do pliku wyjściowego (również w formacie .json), np. w ten sposób:

```
with open('tramwaje_out.json', 'w', encoding='utf-8') as file:
        json.dump(trams, file, ensure_ascii=False)
```

W przykładzie założono, że słownik jest pod nazwą trams. Ponadto, proszę wypisać na ekranie następujące informacje: numer linii – liczba przystanków, posortowane po liczbie przystanków w kolejności malejącej. Na koniec wypisać również **liczbę** (nie nazwy) wszystkich przystanków obsługiwanych przez tramwaje (w tym celu należy znaleźć część wspólną krotek z nazwami przystanków, bo tramwaje często współdzielą ten sam przystanek). **Uwaga**: jako rozwiązanie proszę wysłać zarówno kod programu oraz wyjątkowo również otrzymany plik wynikowy.

2. Napisać funkcję odwracanie(L, left, right) odwracającą kolejność elementów na liście od numeru left do right włącznie. Lista jest modyfikowana w miejscu (in place). Rozważyć wersję iteracyjną i rekurencyjną [Zad. 4.5 https://ufkapano.github.io/algorytmy/lekcja04/zadania.html].

3. Python jest językiem, w którym przy użyciu kodu o niewielkiej długości, ale z umiejętnym użyciem bibliotek, da się osiągnąć interesujące wyniki. Wymaga to jednak zapoznania się z możliwościami różnych bibliotek, celem tego zadania jest właśnie podstawowe użycie biblioteki do rysowania (matplotlib) oraz biblioteki numpy. Zadanie: napisać prosty i zwięzły program, który pozwoli na wczytanie wielomianu funkcji f(x) jako danej wejściowej (łańcuch znakowy) oraz przedział x (od – x_min, do – x_max). Cel: narysować ten wielomian za pomocą `plt.plot(x_val, y_val)` (gdzie `x_val` i `y_val` to będą, odpowiednio, tablica wygenerowana za pomocą `x_val = np.linspace(x_min, x_max, 200)`, a tablica `y_val` wyliczona z użyciem funkcji `eval` dla wartości z tablicy `x_val`. Przykładowo, jeśli wpiszę na wejściu: `5*x**4 + 2*x**3 - x + 6` i podam `-10, 10`, to otrzymam:



Proszę, tak jak na rysunku przykładowym widać, dodać podpis osi X i Y, jakiś tytuł, oraz „grid lines”.

4. Rekurencyjne liczenie ciągu Fibonacciego jest naturalnym algorytmem, niemniej, wyliczanie każdego kolejnego wyrazu ciągu „od początku” jest niepotrzebne. O wiele wydajniejszą metodą byłoby zastosowanie czegoś w rodzaju buforu – pamięci podręcznej, w której zapamiętywalibyśmy poprzednio (wcześniej) wyliczone wyrazy i z nich korzystali. Znacząco przyspieszy to obliczenia. Proszę napisać (uzupełnić poniższy szkielet) kod tak, żeby powstawał słownik – pamięć podręczna z poprzednimi wyliczonymi wartościami i z nich korzystać, a wyliczać nowe tylko gdy jeszcze nie były policzone wcześniej. Słownik proszę zrobić w dekoratorze.

```
import functools

def pamiec(func):
    @functools.wraps(func)
    def wrapper(*args, **kwargs):
    # tu powinien być kod tworzący słownik (element - wartość), który jest sprawdzany
    # do obliczeń wyrazów ciągu - które by były wyliczane rekurencyjnie i wpisywane
    # do słownika tylko gdy wcześniej nie były obliczone
    # normalnie bez buforowania by było return func(*args, **kwargs)

  return wrapper

@pamiec
def fibonacci(n):
    return n if 0 <= n < 2 else fibonacci(n - 1) + fibonacci(n - 2)

for i in range(100):
    print(fibonacci(i))
```

5. W ramach zapoznania się z klasami, proszę napisać klasę o nazwie Bug taką, żeby zawierała licznik wskazujący aktualną liczbę powołanych do życia obiektów, identyfikator (lokalną zmienną obiektu, do której przypiszemy aktualny powiększony licznik). Licznik powinien rosnąć wraz z wywołaniem __init__ oraz maleć z wywołaniem __del__ (uwaga: współdzielony licznik będziemy używać w zapisie Bug.licznik). Proszę też zdefiniować __str__ wypisującą licznik i bieżące id. Proszę też napisać jakiś opisowy komentarz w klasie (w formie *docstring*). Finalnie, niech dla kodu:

```
bugs = []
for i in range(100):
    bugs.append(Bug())
    print(bugs[-1])
```

wypisują się licznik, identyfikator, a przy niszczeniu obiektu w __del__ niech będzie też print informacji typu 'Koniec', licznik, identyfikator.