Podstawy Programowania

Instrukcja do laboratorium 3

1 Zasady ogólne

Ta instrukcja jest przeznaczona na 6 godzin lekcyjnych. Do wykonania i przesłania jest 8 zadań domowych. Ocena za laboratorium zależy od liczby dobrze zrobionych zadań, skala ocen jest pokazana w Tabeli 1.

Tablica 1: Skala ocen.

Liczba zrobionych zadań domowych	Ocena
4	3.0
5	3.5
6	4.0
7	4.5
8	5.0

Kod wszystkich wykonanych programów proszę wkleić do jednego pliku, podpisując poszczególne zadania za pomocą komentarzy. Zadanie proszę wklejać po kolei (1, 2, 3...). Plik z zadaniami proszę przesłać na platformie Moodle. Plik musi mieć nazwę `numeralbumu_lab3.py`. **Plik musi być plikiem tekstowym z rozszerzeniem `.py`**. Wysłanie pracy w nieodpowiedniej formie może skutkować wystawieniem oceny niedostatecznej za te laboratorium.

UWAGA: Termin oddania zadania jest ustawiony w systemie moodle. W przypadku nie oddania zadania w terminie, uzyskana ocena będzie zmniejszana o 0,5 za każdy zaczęty tydzień opóźnienia. Zadania oddawane później niż miesiąc po terminie ustawionym na moodle są oddawane i rozliczane w trybie indywidualnym na zajęciach lub po umówieniu się z prowadzącym.

UWAGA: W przypadku wysłania zadania w formie niezgodnej z opisem w instrukcji prowadzący zastrzega prawo do wystawienia oceny negatywnej za taką pracę. Przykład: wysłanie `.zip` lub `.pdf` tam, gdzie był wymagany plik tekstowy z rozszerzeniem `.py`.

Listing 1: **Przykład dobrze sformatowanego pliku z zadaniami.**

```
# Zadanie 1
a = 4
b = 2
print(a + b)

# Zadanie 2
# Brak

# Zadanie 3
a = 3
b = 2
print(a ** b)

# ...
```

2 Zakres tematyczny

- Wyjątki

  - https://www.w3schools.com/python/python_try_except.asp

- Działanie na plikach
  - https://www.w3schools.in/python-tutorial/file-handling/
  - https://realpython.com/working-with-files-in-python/
  - Zalecane jest użycie wzorcu `with open(...)  as f`
- Funkcje anonimowe `lambda`
  - https://www.w3schools.com/python/python_lambda.asp
- Generatory i słowo kluczowe `yield`
  - https://stackoverflow.com/questions/231767/what-does-the-yield-keyword-do
  - https://www.geeksforgeeks.org/python-yield-keyword/
- Lista zagnieżdżona
  - https://www.learnbyexample.org/python-nested-list/
  - https://www.geeksforgeeks.org/nested-list-comprehensions-in-python/
- Moduł math
  - https://www.w3schools.com/python/python_modules.asp
  - https://docs.python.org/3/library/math.html
  - https://www.w3schools.com/python/module_math.asp
- Moduł itertools
  - https://docs.python.org/3/library/itertools.html
- Moduł collections
  - https://docs.python.org/3/library/collections.html
- Inne możliwości biblioteki standardowej
  - https://docs.python.org/3/library/index.html
- Operatory * i **
  - https://stackoverflow.com/questions/2921847/what-does-the-star-and-doublestar-operator-mean-in-a-fu
    2921893#2921893

3 Zadania do wykonania

3.1 Zadania do wykonania na zajęciach

Te zadania są robione w trakcie zajęć, nie trzeba ich przesyłać na moodlu.

1. Napisz program, który wygeneruje listę składającą się z 50 losowych liczb całkowitych z zakresu od 0 do 10, a następnie wyświetli 5 najczęściej spotykanych liczb. Do wykonania zadania należy użyć funkcji `Counter()`, z modułu `collections`.

   Użyteczne wyrażenia: `collections.Counter(), most_common(), random.randrange()`.

2. Zastosuj funkcje z modułu `itertools`, żeby wypisać na ekran permutacje i kombinacje dwusymbolowe, które można złożyć z liter napisu podanego przez użytkownika.

```
Napis pobrany od użytkownika: "ABCD"
Permutacje: AB AC AD BA BC BD CA CB CD DA DB DC
Kombinacje: AB AC AD BC BD CD
```

Użyteczne wyrażenia: `itertools.permutations()`, `itertools.combinations()`, `input()`.

3. Utwórz strukturę składającą się z list zagnieżdżonych, która będzie reprezentować macierz (w rozumieniu matematycznym). Następnie, zaimplementuj funkcję, która będzie wykonywała mnożenie macierzy przez wektor pionowy (reprezentowany przez zwykłą listę). W przypadku gdy do funkcji jako drugi argument jest podana macierz (lista zagnieżdżona) zamiast listy, należy wyrzucić wyjątek `ValueError` w którym będą umieszczone stosowne informacje. Zademonstruj działanie zaimplementowanej funkcji.

Listing 2: Przykład definiowania macierzy i przykładowy wynik działania programu.

```
# Przykładowa macierz:
a = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
# Przykładowy wektor:
b = [6, 4, 2]
# Wynik mnożenia macierzy a przez pionowy wektor b:
[20, 56, 92]
```

Użyteczne wyrażenia: `def`, `return`, `for`, `range()`, `ValueError()`.

4. Wczytaj plik `points.txt`, który zawiera współrzędne 100 punktów położonych w przestrzeni dwuwymiarowej (jeden wiersz - jeden punkt, współrzędne x i y są oddzielone od siebie spacjami). Następnie, wypisz na ekran dwa punkty które są położone najbliżej siebie.

   Użyteczne wyrażenia: `with`, `open()`, `read()`, `splitlines()`, `math.dist()`, `itertools.combinations()`.

5. Napisz funkcję, która służy do rozwiązywania równań kwadratowych przyjmującą trzy argumenty, będące współczynnikami równania. Do obliczenia pierwiastka z delty należy użyć funkcji `math.sqrt()`. **Proszę nie używać warunków dla sprawdzenia czy delta jest ujemna, a wyjątek wyrzucany przez funkcje `math.sqrt()` przechwycić poza funkcją.**

   Następnie, napisz program który będzie w pętli odpytywać użytkownika o kolejne równania które trzeba rozwiązać (użytkownik ma podawać trzy współczynniki w jednym wierszu, lub podawać kolejne współczynniki w kolejnych wierszach, dane pobierać do momentu wprowadzenia przez użytkownika pustego wierszu). Znalezione pierwiastki należy wypisać na ekran, jeżeli równanie nie ma pierwiastków rzeczywistych, odpowiednio poinformować użytkownika o tym. Proszę również pamiętać, że jeżeli funkcja została zaimplementowana zgodnie z wymaganiami będzie ona wyrzucać wyjątek, który należy odpowiednio obsłużyć.

   Użyteczne wyrażenia: `input()`, `math.sqrt()`, `try`, `expect`.

6. Zaimplementuj funkcje, która utworzy listę liczb pierwszych do n (n jest argumentem funkcji) korzystając z algorytmu sita Eratostenesa. Następnie, napisz program, który wygeneruje za pomocą tej funkcji listę liczb pierwszych do 100 i zapisze ją do pliku `prime_numbers.txt`, zapisując po 5 kolejnych liczb pierwszych w każdym wierszu pliku wynikowego.

   Więcej o sicie Eratostenesa: https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes.

   Użyteczne wyrażenia: `with`, `open()`, `write()`.

7. Zaimplementuj funkcje-generator, której zadaniem będzie generowanie kolejnych n liczb ciągu Fibonacciego (wzór jest podany niżej). Liczba n jest podawana jako argument funkcji. Funkcja ma używać słowa kluczowego `yield`. Przy implementacji nie używamy rekurencji!

$$a_n = \begin{cases} 0, & n = 1 \\ 1, & n = 2 \\ a_{n-1} + a_{n-2}, & n > 2 \end{cases} \tag{1}$$

   Użyteczne wyrażenia: `yield`.

8. Wczytaj plik `students.csv`, który zawiera w kolejnych wierszach dane o studentach w postaci imię, nazwisko, numer albumu, ocena, oddzielone przycinkami. Należy wczytać te dane do struktury listy krotek (list of tuples), przykładowe wczytane dane są pokazane w poniższym przykładzie. Następnie, należy użyć funkcji wbudowanych `sorted(), max()` z argumentem `key` i wyświetlić:

   - Dane o studentach posortowane po numerach albumów rosnąco,
   - Dane o studentach posortowane po ocenach malejąco,
   - Dane studenta z najwyższą oceną.

Listing 3: Przykładowe dane po wczytaniu.

```
[
    ('Jan', 'Kowalski', 1523, 4),
    ('Jan', 'Nowak', 4125, 4.5),
    ...
]
```

Użyteczne wyrażenia: `sorted(), max(), lambda, with, open(), read(), splitlines(), split()`.

3.2 Zadania domowe

Zadania przeznaczone do samodzielnego zrobienia w domu. Rozwiązania tych zadań należy przesłać na moodlu.

1. Napisz program, który wygeneruje listę składającą się z 50 losowych liczb całkowitych z zakresu od 0 do 10, a następnie wyświetli 5 najczęściej spotykanych liczb. Do wykonania zadania należy użyć funkcji `Counter()`, z modułu `collections`.

   Użyteczne wyrażenia: `collections.Counter(), most_common(), random.randrange()`.

2. Zastosuj funkcje z modułu `itertools`, żeby wypisać na ekran permutacje i kombinacje trzysymbolowe, które można złożyć z liter napisu podanego przez użytkownika.

```
Napis pobrany od użytkownika: "ABC"
Permutacje: ABC ACB BAC BCA CAB CBA
Kombinacje: ABC
```

   Użyteczne wyrażenia: `itertools.permutations(), itertools.combinations(), input()`.

3. Zaimplementują funkcję służąca do obliczenia wyznacznika macierzy kwadratowych o rozmiarze 2. Macierz powinna być jedynym argumentem tej funkcji. W razie podania macierzy o złych wymiarach należy wyrzucić wyjątek `ValueError` z odpowiednią informacją.

Listing 4: Przykład definiowania macierzy i przykładowy wynik działania programu.

```
Macierz:
a = [
    [1, 2],
    [3, 4],
]
Wyznacznik: -2
```

   Użyteczne wyrażenia: `def, return, if, else, raise, ValueError()`.

4. Wczytaj plik `points.txt`, który zawiera współrzędne 100 punktów położonych w przestrzeni dwuwymiarowej (jeden wiersz - jeden punkt, współrzędne x i y są oddzielone od siebie spacjami). Pobierz od użytkownika dwie liczby będącymi współrzędnymi x, y kolejnego punktu. Następnie, wypisz na ekran współrzędne 10 punktów które są położone najbliżej tego wczytanego od użytkownika punktu.

   Użyteczne wyrażenia: `with, open(), read(), write(), math.dist()`.

5. Napisz funkcję, która służy do rozwiązywania równań kwadratowych przyjmującą trzy argumenty, będące współczynnikami równania. Do obliczenia pierwiastka z delty należy użyć funkcji `math.sqrt()`. **Proszę nie używać warunków dla sprawdzenia czy delta jest ujemna, a wyjątek wyrzucany przez funkcje `math.sqrt()` przechwycić poza funkcją.**

Następnie, napisz program który wczytuje zawartość pliku `equations.txt` (jeden wiersz - jedno równanie, trzy współczynniki oddzielone spacjami). Użyj wcześniej zaimplementowanej funkcji do rozwiązywania 50 równań znajdujących się we wczytanym pliku. Znalezione pierwiastki należy zapisać w nowym pliku pod nazwą `equations_results.txt` w formacie: jeden wiersz - dwa pierwiastki jednego równania, oddzielone spacjami, jeżeli równanie nie ma pierwiastków rzeczywistych należy zostawić w pliku wiersz pusty. Proszę również pamiętać, że jeżeli funkcja została zaimplementowana zgodnie z wymaganiami będzie ona wyrzucać wyjątek, który należy odpowiednio obsłużyć.

Użyteczne wyrażenia: `with, open(), read(), splitlines(), write(), math.sqrt(), try, expect`.

6. Zaimplementuj funkcję, sprawdzającą czy liczba n podana jako argument jest liczbą pierwszą. Funkcja ma sprawdzać, czy liczba podana jako argument jest podzielna na potencjalne dzielniki tej liczby. Implementacja ma używać następujące usprawnienia:

- liczba 1 i wszystkie liczby ujemne nie są pierwsze,
- liczby parzyste większe od 2 mogą być wyeliminowane od razu,
- dla pozostałych liczb wystarczy sprawdzić ich podzielność na liczby nieparzyste od 3 do $\sqrt{n}$ (włącznie!).

Następnie, napisz program, który znajdzie które z pierwszych 30 liczb ciągu Fibonacciego są też liczbami pierwszymi. Liczby z ciągu Fibonacciego mogą być generowane w dowolny sposób.

Użyteczne wyrażenia: `math.sqrt(), range()`.

7. Zaimplementuj funkcje-generator, używającą słowo kluczowe `yield` która będzie służyć do generowania kolejnych wyrazów ciągu Collatz'a zaczynając od podanej jako argument funkcji liczby n do 1. Przy implementacji nie używamy rekurencji! Ciąg Collatz'a jest zdefiniowany następującym wzorem:

$$c_{n+1} = \begin{cases} \frac{1}{2}c_n & gdy\ c_n\ jest\ parzysta \\ 3c_n + 1 & gdy\ c_n\ jest\ nieparzysta \end{cases} \tag{2}$$

Listing 5: Przykładowe wywołanie funkcji-generatora.

```
>>> for i in collatz(13):
...     print(i, end=" -> " if i != 1 else "\n")
...
13 -> 40 -> 20 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1
```

Użyteczne wyrażenia: `yield`.

8. Wczytaj plik `cars.csv`, który zawiera w kolejnych wierszach dane o samochodach w postaci producent, model, rok, cena, oddzielone przycinkami. Należy wczytać te dane do struktury listy krotek (list of tuples), przykładowe wczytane dane są pokazane w poniższym przykładzie. Następnie, należy użyć funkcji wbudowanych `sorted(), min()` z argumentem `key` i wyświetlić:

- Dane o samochodach, posortowane od najtańszego do najdroższego,
- Dane o samochodach, posortowane od najnowszego do najstarszego,
- Dane o najtańszym samochodzie.

Listing 6: Przykładowe dane po wczytaniu.

```
[
    ('Volkswagenn', 'Golf', 2006, 15000),
    ('Honda', 'Civic', 2004, 10000),
    ...
]
```

Użyteczne wyrażenia: `sorted(), min(), lambda, with, open(), read(), splitlines(), split()`.