

Dokumentacja Problemu Filozofów Uczestniczących w Posiłkach

Ta dokumentacja dostarcza przeglądu symulacji problemu filozofów w języku Python przy użyciu bibliotek threading oraz tkinter. Program symuluje zachowanie filozofów siedzących przy stole, z każdym reprezentowanym przez osobny wątek. Filozofowie na zmianę myślą i jedzą, przy czym muszą używać widelców do jedzenia. Jednakże, mogą jedynie jeść, gdy posiadają oba lewy i prawy widelec.

Spis Treści

1. Wymagania
2. Przegląd implementacji
3. Zmienne globalne
4. Interfejs graficzny użytkownika
5. Funkcja filozofa i aktualizacji GUI
6. Generowanie wykresu liniowego
7. Funkcja aktualizacji czasu
8. Wątki filozofów
9. Wykonanie programu
10. Przykładowe symulacje
11. Wnioski

Wymagania

- Python 3.x
- Biblioteka tkinter (standardowa biblioteka GUI w Pythonie)
- Biblioteka threading (standardowa biblioteka do wątków w Pythonie)
- Pyplot – narzędzie do generowania wykresów
- CSV – narzędzie do zapisywania danych do łatwych w zarządzaniu plikami .csv

Przegląd Implementacji

Program tworzy GUI za pomocą biblioteki tkinter, aby wizualnie przedstawiać filozofów i ich działania. Każdy filozof jest reprezentowany przez osobny wątek, a korzystają oni z ustalonej liczby widełek. Program działa przez określony czas (TIME_TO_RUN) i symuluje działania filozofów w tym okresie. Bibliotekę tkinter instalujemy w terminalu przy użyciu polecenia „`pip install tkinter`”

Zmienne Globalne

`NUM_PHILOSOPHERS`: Liczba filozofów przy stole.

TIME_TO_RUN: Całkowity czas (w sekundach) trwania symulacji.

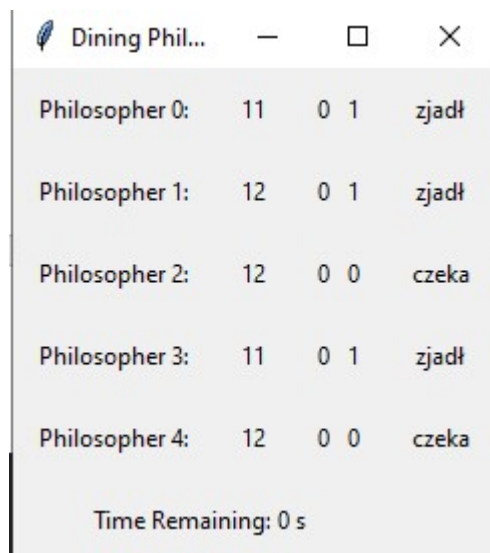
forks: Lista obiektów threading.Semaphore reprezentujących widełki na stole. **eat_count**:

Lista do śledzenia liczby posiłków zjedzonych przez każdego filozofa.

fork_status: Lista wskazująca status widełek każdego filozofa (0: brak widelca, 1: własny widelec, 2: pożyczony widelec).

Interfejs Graficzny Użytkownika (GUI)

GUI jest tworzone przy użyciu biblioteki tkinter. Składa się z etykiet wyświetlających informacje o każdym filozofie, w tym ich ID, liczbie zjedzonych posiłków oraz statusie widełek oraz aktualny stan. Są trzy stany: „zjadł”, „je” i „myśli”. GUI zawiera również etykietę wyświetlającą pozostały czas symulacji.



The screenshot shows a window titled "Dining Phil..." with standard window controls. It contains a table with 5 rows representing philosophers. Each row has 5 columns: Philosopher ID, a number, two status indicators (0 or 1), and a text label. At the bottom, there is a label "Time Remaining: 0 s".

Philosopher 0:	11	0	1	zjadł
Philosopher 1:	12	0	1	zjadł
Philosopher 2:	12	0	0	czeka
Philosopher 3:	11	0	1	zjadł
Philosopher 4:	12	0	0	czeka

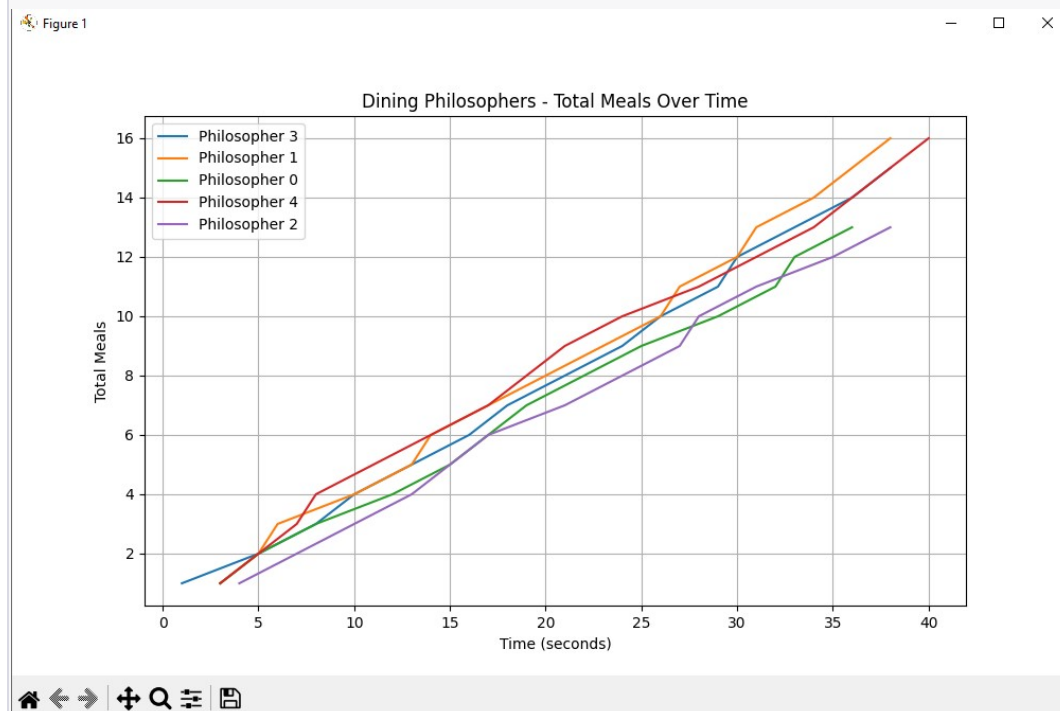
Time Remaining: 0 s

Funkcja Filozofa i Funkcja Aktualizacji GUI

Funkcja **philosopher** reprezentuje zachowanie każdego filozofa jako osobny wątek. Filozof na zmianę myśli i próbuje jeść. Używa semaforów (, aby **forks**) zarządzać dostępem do widełek i aktualizować GUI zgodnie z tymi zmianami. Funkcja **update_gui** aktualizuje GUI, aby odzwierciedlić zmiany w informacjach o filozofie, takich jak ID, liczba zjedzonych posiłków oraz status widełek.

Generowanie Wykresu Liniowego

Funkcja `generate_line_chart` wczytuje dane z pliku CSV i generuje wykres liniowy przedstawiający ilość spożytych posiłków w czasie dla każdego filozofa.



Funkcja Aktualizacji Czasu

Funkcja `update_time` aktualizuje pozostały czas w GUI. Jest wywoływana okresowo za pomocą metody `after`, aby zapewnić aktualizację w czasie rzeczywistym.

Wątki Filozofów

Program tworzy listę wątków filozofów (`philosopher_threads`) i uruchamia każdy wątek jako daemona. Wątki uruchamiają funkcję `philosopher` z ich odpowiednim ID filozofa.

Wykonanie Programu

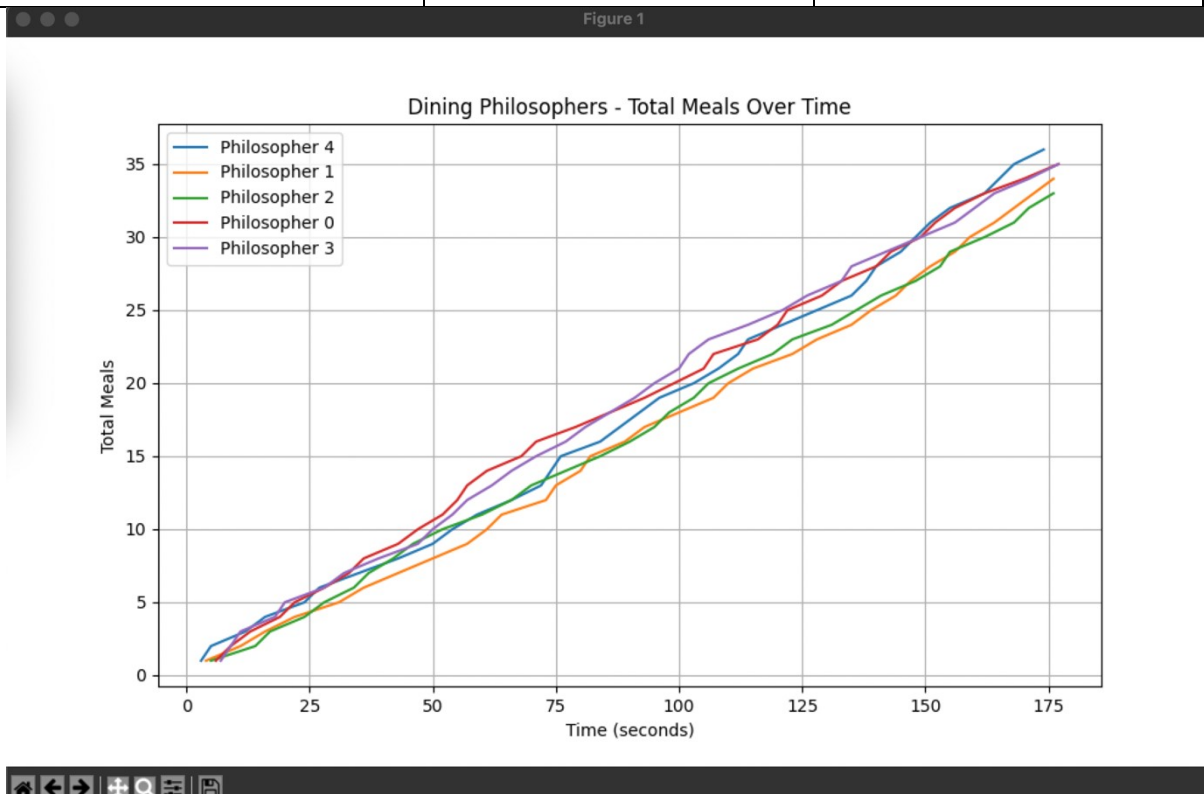
Program inicjalizuje zmienne globalne, tworzy GUI, uruchamia wątki filozofów i aktualizuje czas. GUI działa do momentu upływu określonego czasu (`TIME_TO_RUN`), a następnie program kończy działanie.

Uwaga: Symulacja może prowadzić do różnych liczb zjedzonych posiłków przez filozofów w zależności od czasów oczekiwania na uzyskanie widełek.

Dane z przykładowych symulacji

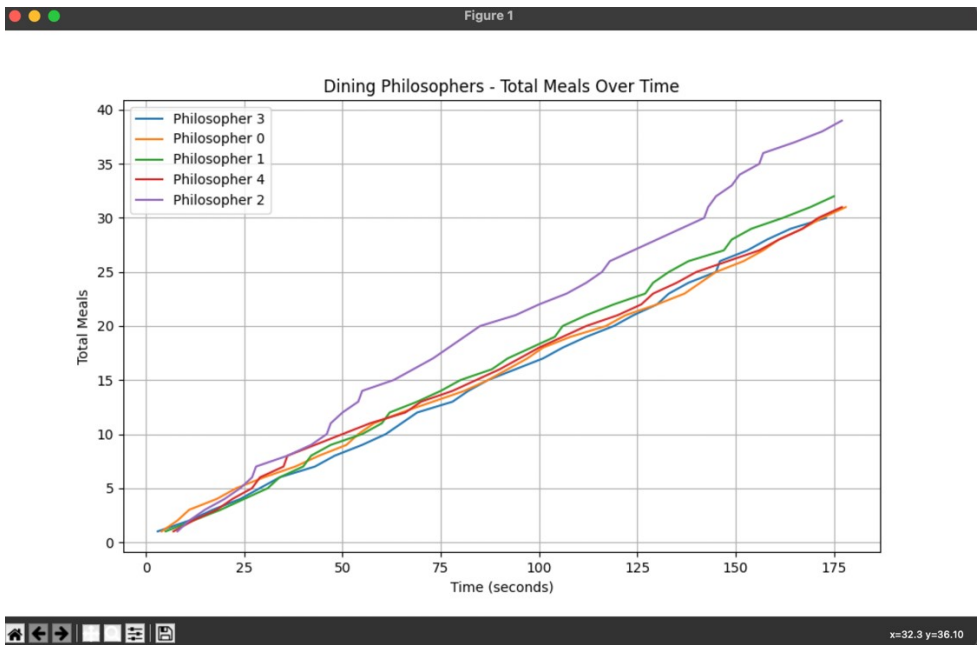
1. Symulacja

	od	do
Myślenie	2s	4s
Jedzenie	1s	5s



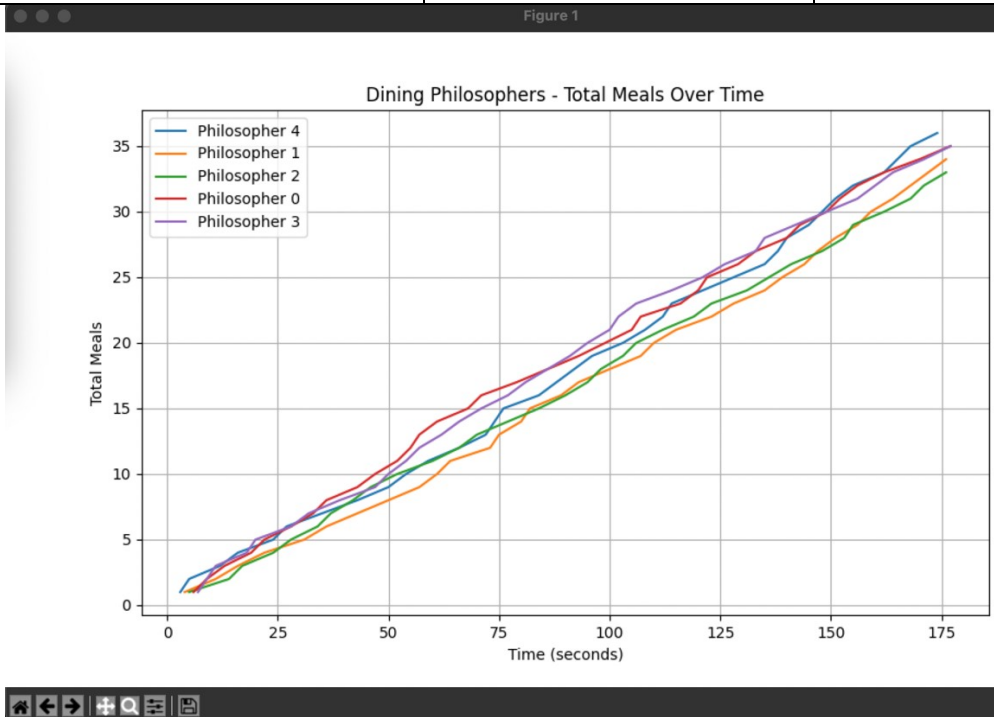
2. Symulacja

	od	do
Myślenie	1s	6s
Jedzenie	2s	3s



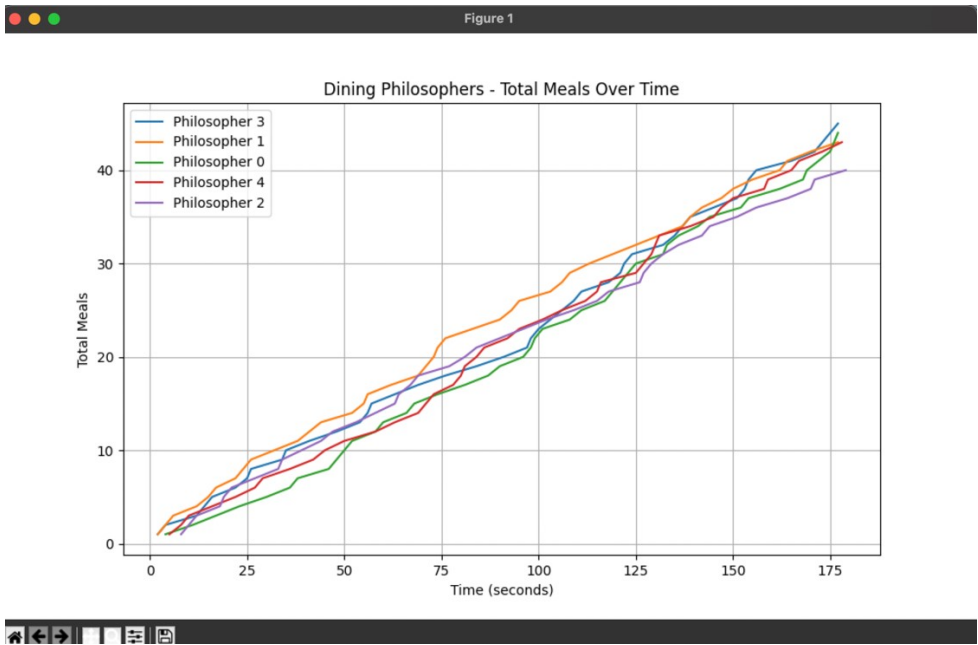
3. Symulacja

	od	do
Myślenie	2s	4s
Jedzenie	1s	5s



4. Symulacja

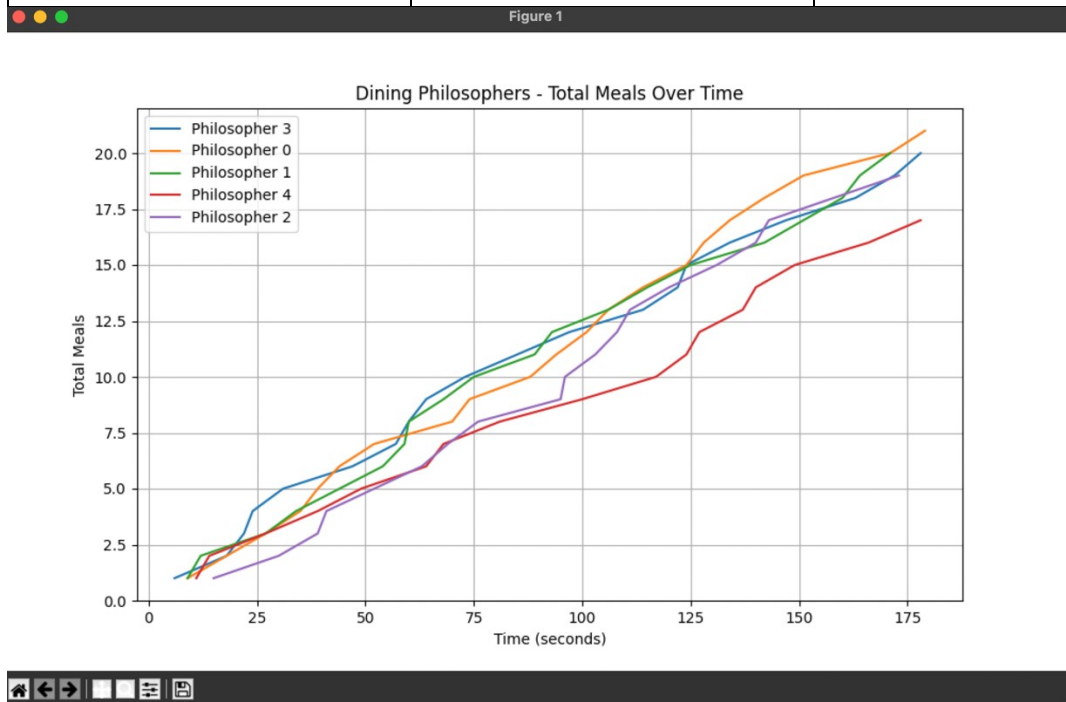
	Od	Do
Myślenie	1s	3s
Jedzenie	3s	6s



5. Symulacja

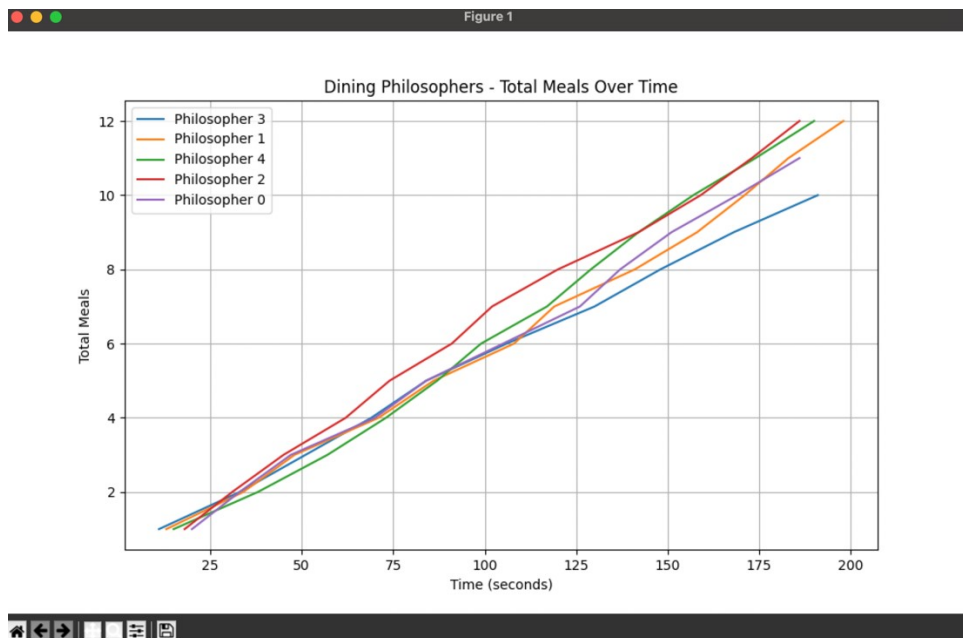
	Od	Do
Myślenie	1s	10s
Jedzenie	1s	10s

Figure 1



6. Symulacja

	Od	Do
Myślenie	10s	15s
Jedzenie	1s	10s



Wnioski

1. **Losowość wpływa na ilość spożytych posiłków:** Filozofowie jedzą losowo, co jest odzwierciedlone w kodzie przez losowy czas oczekiwania na jedzenie (funkcja `time.sleep(random.uniform(1, 10))`). To sprawia, że różni filozofowie spożywają posiłki w różnym tempie.
2. **Czekanie na widelce wpływa na tempo jedzenia:** Filozofowie muszą czekać na dostęp do obu widelców przed jedzeniem. To oczekiwanie wpływa na różnice w ilości spożywanych posiłków, ponieważ nie zawsze każdy filozof będzie mógł zacząć jedzenie natychmiast po zakończeniu myślenia.
3. **Dostępność widelców wpływa na tempo jedzenia:** Modeluje się to za pomocą semaforów (forks). Jeśli filozofowi uda się zająć oba widelce, zacznie jeść. Jednakże, jeśli drugi widelec jest zajęty przez innego filozofa, musi poczekać, co wpływa na jego tempo jedzenia.
4. **Czas trwania symulacji wpływa na ilość spożytych posiłków:** Symulacja jest ograniczona czasowo (`TIME_TO_RUN`). Filozofowie kończą jedzenie po pewnym czasie, co wpływa na ogólną ilość spożywanych posiłków.
5. **Wpływ losowości na czas myślenia:** Również czas myślenia jest losowy (`time.sleep(random.uniform(10, 15))`), co wpływa na to, kiedy dany filozof zacznie ponownie próbować jeść.
6. **Wyświetlanie danych w czasie rzeczywistym:** Interfejs graficzny (tkinter) pozwala na monitorowanie aktualnego stanu filozofów, ich myślenia, oczekiwania na widelce, jedzenia i zakończenia jedzenia. To pozwala na wizualizację i zrozumienie, co się dzieje w trakcie symulacji.