1 Małe problemy

Zaczniemy od przeanalizowania pewnych prostych zadań, do rozwiązania których wystarczy kilka dość krótkich funkcji. Mimo że są to małe problemy, pozwalają na poznanie pewnych interesujących technik rozwiązywania. Potraktuj je jak rozgrzewkę.

1.1. Ciąg Fibonacciego

Ciąg Fibonacciego to ciąg takich liczb, z których każda – z wyjątkiem pierwszej i drugiej – jest sumą dwóch poprzednich:

```
0, 1, 1, 2, 3, 5, 8, 13, 21...
```

Pierwsza liczba Fibonacciego w ciągu ma wartość `0`. Czwarta liczba Fibonacciego to `2`. Wynika z tego, że aby otrzymać wartość dowolnego wyrazu ciągu Fibonacciego, oznaczonego jako `n`, możemy użyć następującego wzoru:

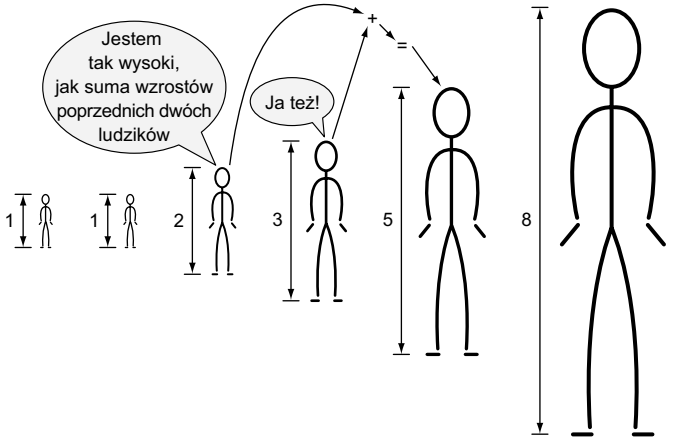
```
fib(n) = fib(n - 1) + fib(n - 2)
```

1.1.1. Pierwsza próba rekurencyjna

Powyższy wzór do obliczania wartości wyrazu ciągu Fibonacciego (rys. 1.1) stanowi pewien rodzaj pseudokodu, który możemy w naiwny sposób przetłumaczyć do *rekurencyjnej* funkcji Pythona (funkcja rekurencyjna to taka, która wywołuje samą siebie). To mechaniczne tłumaczenie będzie naszą pierwszą próbą napisania funkcji, która zwraca wybraną wartość ciągu Fibonacciego.

Listing 1.1. fib1.py

```
def fib1(n: int) -> int:
    return fib1(n - 1) + fib1(n - 2)
```



Rysunek 1.1. Wzrost każdego z ludzików wynosi tyle, ile suma wzrostów poprzednich dwóch ludzików

Spróbujmy uruchomić tę funkcję, wywołując ją z określoną wartością.

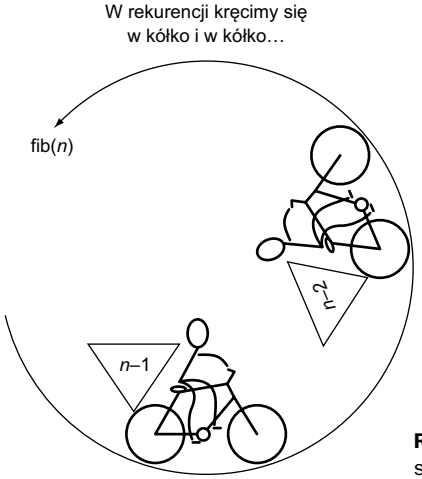
Listing 1.2. fib1.py kontynuacja

```
if __name__ == "__main__":
    print(fib1(5))
```

Ojej! Uruchomienie fib1.py generuje błąd rekurencji:

```
RecursionError: maximum recursion depth exceeded
```

Problem polega na tym, że funkcja `fib1()` będzie działała w nieskończoność bez zwrócenia końcowego wyniku. Każde wywołanie `fib1()` skutkuje dwoma kolejnymi wywołaniami `fib1()`, a końca nie widać. Taką sytuację nazywamy *nieskończoną rekurencją* (rys. 1.2) i jest ona analogiczna do *nieskończonej pętli.*



Rysunek 1.2. Funkcja rekurencyjna `fib(n)` wywołuje samą siebie z argumentami `n-2` i `n-1`