Laboratorium 1: Krótkie wprowadzenie do czasowej złożoności obliczeniowej

1. **Co to złożoność obliczeniową? Jak ją badamy?**
   Czasowa złożoność obliczeniowa mówi nam "jak szybko" wykonuje się dany algorytm (program) w zależności od wielkości danych wejściowych. Dla przypomnienia każdy algorytm składa się z 3 części:
   Wejście: Dane wejściowe i ich rozmiar n, np. tablica liczb naturalnych i jej długość n
   Ciało algorytmu: ciąg instrukcji, np. szukanie maksymalnego elementu tablicy
   Wyjście: rezultat obliczeń, np. wypisanie na ekranie największego elementu tablicy

```
zbior = [4, 3, 20, 34, 12, 10]
def maksimum(iloscElementow):
    maks = zbior[0]
    for i in range(1, iloscElementow):
        if maks < zbior[i]:
            maks = zbior[i]
    return maks
```

   Realny czas wykonywania algorytmu:

$$\text{T(n)} = \text{n} * (\text{czas porównywania elementów} + \text{czas zamian})$$

   Złożoność problemu, czyli ile kroków (podstawowych instrukcji, zależnych m.in. od szybkości procesora) wykonuje dany algorytm:

$$\text{T(n)} = \text{n}$$

2. **Dlaczego zajmujemy się złożonością czasową?**
   Przykłady złożoności dla różnych n

	złożoność - $f(n)$				
n	n	$n * log_2(n)$	n^2	2^n	$n!$
10	10	≈ 34	100	1024	3628800
100	100	≈ 650	10000	1.268E30	∞
1000	1000	≈ 10000	1000000	1.072E301	∞