

## Zestaw 1

1. Napisać program, który czyta podane jako zewnętrzne argumenty liczby naturalne, a następnie każdą rozkłada na czynniki pierwsze (co polega na zapisaniu dowolnej liczby naturalnej za pomocą iloczynu liczb pierwszych). Wymagany jest format wyjściowy w postaci  $a_1^{k_1} \cdot a_2^{k_2} \cdot \dots \cdot a_n$ , jeśli  $k_i=1$  to opuszczamy wykładnik potęgi. Przykładowo, jeśli wywołamy:

```
zadanie1.py 4407 13041599400
```

to powinno się wypisać (proszę tak to sformatować):

```
4407 = 3*13*113
13041599400 = 2^3*3^4*5^2*805037
```

Do wczytania zewnętrznych argumentów proponuję na początek coś bardzo podobnego do tego, co jest w języku C++, czyli użycie listy argumentów (bez używania getopt czy argparse):

```
import sys # importujemy modul

argv = sys.argv[1:] # argv to lista, a 1: robi selekcje bez pierwszego argumentu - nazwy programu

for i in range(1,len(argv)): # za pomoca generatora
    print(int(sys.argv[i])) # wpisujemy, rzutowanie z typu str na int przyda sie potem
```

Opis i program w C++ <https://www.algorytm.edu.pl/algorytmy-maturalne/rozklad-na-czynniki.html>

Pomocny może też być kalkulator rozkładu na czynniki pierwsze <https://www.liczebnik.pl/czynniki-pierwsze.php>

2. Napisać program rysujący „miarkę” o zadanej długości. Należy prawidłowo obsłużyć liczby składające się z kilku cyfr (ostatnia cyfra liczby ma znajdować się pod znakiem kreski pionowej). Należy zbudować pełny string, a potem go wypisać. [Zad. 3.5 <https://ufkapano.github.io/algorytmy/lekcja03/zadania.html>]

```
|.....|.....|.....|.....|.....|.....|.....|.....|.....|.....|.....|.....|
0      1      2      3      4      5      6      7      8      9      10     11     12
```

3. Napisać program, który będzie wyświetlał bieżący czas (tak ma to wyglądać: ► 14:48:31 ◀), aktualizowany dynamicznie. Czas można odczytać na wiele sposobów, użyjmy moduł `datetime`, wtedy, bieżący punkt w czasie dostaniemy: `now = datetime.now()` i za pomocą składowych `now.hour`, `now.minute`, `now.second` mamy potrzebne wartości. Przy czym dla sekund należy sprytnie podmienić sekundy w zakresie 0..9 tak, żeby przed nimi wyświetlało się zero (np. nie 5, tylko 05). Znaczkę na początku i końcu mają kod `chr(16)` i `chr(17)`. Zegar musi być wyświetlany w nieskończonej pętli funkcją `print()`, argument `end='\r'` zapewni nadpisywanie. Potrzebne jest jeszcze (z modułu `time`)wołanie czegoś typu `time.sleep(0.5)` w pętli, żeby niepotrzebnie nie odświeżać zbyt często bieżącego odczytu czasu.
4. Napisać program, który dynamicznie wyświetla „pasek postępu” o zadanej (zdefiniowanej parametrem) długości. Powinno to wyglądać tak (kolejne etapy):
 

```
|-----| 0%
|=====| 56%
|=====| 100%
```
5. Napisz program, w którym dowolny tekst " Hello world! " przesuwają się w terminalu w pionie: w dół oraz w jakimś miejscu odbija się i do góry, aż do krawędzi okienka itd.