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*a_2^k_2*…*a_3, jeśli k_i==1 to opuszczamy wykładnik potęgi. Przykładowo, jeśli wywołamy:

```
zadanie1.py  4407  13041599400
```

to powinno się wypisać (proszę tak to sformatować, sprawdzany będzie rozkład czyli prawa strona):

```
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

Wymagania formalne Użyć plik ZADANIE1/zadanie1.py w repozytorium GitHub Classroom do uzupełnienia swoim kodem. Nie zmieniać nazw funkcji. Dane wejściowe w postaci liczb naturalnych, oddzielonych spacją (spacjami). Testowane będzie działanie funkcji `rozkład_na_czynniki(n)`, która powinna zwracać uformowany łańcuch znakowy w postaci jak poniżej „wyjście".
wejście: 805037 **wyjście:** 805037
wejście: 13041599400 **wyjście:** 2^3*3^4*5^2*805037

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
```

Wymagania formalne Użyć plik ZADANIE2/zadanie2.py w repozytorium GitHub Classroom do uzupełnienia swoim kodem. Nie zmieniać nazwy funkcji. Długość miarki podana jest jako zmienna w programie, np. dlugosc_miarki = 123. Testowane będzie działanie funkcji `rysuj_miarke()`,czy zwrócony przez nią string jest poprawną miarką. Badane będą przypadki od 0 (tylko jedna pionowa kreska) aż do 999.

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). Znaczki 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.

Wymagania formalne Użyć plik ZADANIE3/zadanie3.py w repozytorium GitHub Classroom do uzupełnienia swoim kodem. Program będzie uruchomiony i oceniony wizualnie, ale jedyny test sprawdzi, czy obecna jest w kodzie funkcja `wyswietl_zegar()`.

4. Napisać program, który dynamicznie wyświetla „pasek postępu” o zadanej (zdefiniowanej parametrem) długości. Powinno to wyglądać tak (kolejne etapy):

```
|--------------------------------------------------| 0%
|==========================----------------------| 56%
|==================================================| 100%
```

Wymagania formalne Użyć plik ZADANIE4/zadanie4.py w repozytorium GitHub Classroom do uzupełnienia swoim kodem. Program będzie uruchomiony i oceniony wizualnie, ale jedyny test sprawdzi, czy obecna jest w kodzie funkcja `pasek_postepu(n)`.

5. Napisz program, w którym dowolny tekst `" Hello world! "` przesuwa się w terminalu w pionie: w dół oraz w jakimś miejscu odbija się i do góry, aż do górnej krawędzi okienka itd.

Wymagania formalne Użyć plik ZADANIE5/zadanie5.py w repozytorium GitHub Classroom do uzupełnienia swoim kodem. Program będzie uruchomiony i oceniony wizualnie, ale jedyny test sprawdzi, czy obecna jest w kodzie funkcja `przesun_tekst_w_pionie(txt, n)`.