Zadanie 1. Napisz klasę `Ulamek` reprezentującą ułamek zwykły posiadającą prywatne pola `licznik` i `mianownik`. Dodaj:

- konstruktor przyjmujący wartości licznika i mianownika (o domyślnych wartościach odpowiednio 0 i 1),
- metody `getLicznik` i `getMianownik` zwracające wartość licznika i mianownika,
- metodę `mnoz` mnożącą dwa ułamki i zwracającą wynik,
- metodę `toFloat` konwertującą ułamek na liczbę zmiennoprzecinkową,
- metody `__str__` i `__repr__` zwracające ułamek jako ciąg znaków.

Utwórz dwa ułamki, wypisz je po konwersji na ciągi znaków (użyj funkcji `str` i `repr`) i na liczby zmiennoprzecinkowe oraz wypisz wynik mnożenia ich przez siebie.

`https://docs.python.org/3/tutorial/classes.html#a-first-look-at-classes`

Zadanie 2. Utwórz klasę `SuperUlamek` dziedziczącą po klasie `Ulamek`. Dodaj do nowej klasy metodę `uprosc` upraszczającą dany ułamek. Nadpisz metodę `mnoz` z klasy bazowej tak, aby po przemnożeniu ułamek był uproszczony. Do wykonania samego mnożenia wykorzystaj funkcję `super` aby wywołać metodę `mnoz` z klasy bazowej. Przetestuj nową klasę wykonując działanie $\frac{10}{21} \times (-\frac{35}{6})$.

Pamiętaj o wykorzystaniu funkcji `super` w konstruktorze klasy `SuperUlamek`.

`https://docs.python.org/3/tutorial/classes.html#inheritance`

`http://en.wikipedia.org/wiki/Greatest_common_divisor`

Zmodyfikuj klasę tak, aby można było mnożyć ułamki zwykłym operatorem mnożenia (`https://docs.python.org/3.5/library/operator.html`).

Zadanie 3. Napisz implementacje następujących klas:

- `Obiekt` z funkcją `rysuj` wypisującą na konsolę tekst "[nieznany obiekt]",
- `Prostokat` dziedziczącą po klasie `Obiekt`, nadpisującą funkcję `rysuj` tak, aby rysowała prostokąt którego długości boków były podanym w konstruktorze,
- `Choinka` dziedziczącą po klasie `Obiekt`, nadpisującą funkcję `rysuj` tak, aby rysowała choinkę o podanej w konstruktorze liczbie poziomów (patrz Rys. 1). Można w tym celu wykorzystać mnożenie ciągu znaków przez liczbę (`'*' * 3`) i funkcję `center`,
- `ChoinkaZPrezentami` dziedziczącą po klasie `Choinka`, nadpisującą funkcję `rysuj` tak, aby rysowała choinkę (jak w klasie `Choinka`) oraz poniżej trzy kwadraty o boku 2, jeden obok drugiego. Wykorzystaj funkcję `super`.

```
   *
  ***
   *
  ***
 *****
   *
  ***
 *****
*******
   *
```

Rysunek 1: Choinka o trzech segmentach.

Wykonaj poniższy kod i przeanalizuj jego działanie:

```
a = [Obiekt(), Prostokat(4, 5), Choinka(4), ChoinkaZPrezentami(3)]
for x in a:
    x.rysuj()
    print()
```

Zadanie 4. Dodaj do klasy `Obiekt` z zadania trzeciego pole klasy o nazwie `wypelnienie` oznaczające znak wykorzystywany do rysowania obiektów. Popraw odpowiednio funkcje rysujące. Ustaw nowo dodane pole na `'o'`.

`https://docs.python.org/3/tutorial/classes.html#class-and-instance-variables`

Zadanie 5. Zmodyfikuj kod rysujący obiekty z listy z zadania trzeciego tak, aby:

1. Rysowane były tylko choinki (funkcja `type`).
2. Rysowane były tylko choinki i choinki z prezentami (dwa warianty: z wykorzystaniem funkcji `instanceof` oraz z wykorzystaniem funkcji `type` i `issubclass`).

Zadanie 6.

Przyjrzyj się poniższemu fragmentowi kodu i sprawdź jak działa.

```
class Kwadraty:
    def __init__(self):
        self.num = 0

    def __iter__(self):
        return self

    def __next__(self):
        self.num += 1
        return (self.num-1)**2

for i in Kwadraty():
    print(i)
    if i > 10000:
        break
```

Zmodyfikuj iterator `Kwadraty` tak, aby zaczynał liczyć nie od zera, lecz od liczby przekazanej do konstruktora klasy. Wydziel wyrażenie zwracane w metodzie `__next__` do osobnej metody o nazwie `f`. Zmodyfikuj metodę `f` tak, aby zwracała nie kwadrat liczby, ale `(n%4)`-ty element listy `['a', 'b', 'c', 'd']` (gdzie `n` jest argumentem metody `f`).

Iteratory będą szerzej omawiane na kolejnych zajęciach.

`https://docs.python.org/3/tutorial/classes.html#iterators`