

- Pierwsze listy zadań (1, 2, 3, ~4/4A) dotyczyły głównie problemów arytmetycznych.
- Potem pojawiły się narzędzia, pozwalające na wyjście poza arytmetykę.
- Część z nich to biblioteki (głównie matplotlib, math).
- Ważniejsza część – przegląd podstawowych typów obiektów w Pythonie.
- Z początku znane nam były proste typy danych – liczby i wartości logiczne. Potem bardziej skomplikowane – napisy i kolekcje (listy, krotki, zbiory, słowniki).
- Bardziej specyficzne obiekty, np. pliki.

Obiekt, ogólnie: zestaw danych wraz z operacjami na nich.

Programowanie obiektowe – paradygmat programowania, w którym obiekt jest centralnym konceptem. Modelujemy rzeczywistość wyodrębniając z niej obiekty, opisując operacje na nich, relacje i komunikację między nimi, a następnie reprezentując je w programie.

Takimi wyodrębnionymi obiektami może być cokolwiek:

- W matematyce: liczby, macierze, wektory (i działania na nich), funkcje, przestrzenie, ...
- W przeglądarce: zakładka, strona, przycisk (ogólnie: elementy interfejsu graficznego), ...
- W USOSie: osoba, przedmiot, sprawdzian, zestaw ocen, protokół zaliczeń, ...

Obiekty i programowanie obiektowe

Znane nam typy obiektów w Pythonie (słowniki etc.) są dość uniwersalne, ale bardzo ogólne.

Kluczowa w programowaniu obiektowym [w Pythonie] jest możliwość definiowania *własnych* typów obiektów (wraz z operacjami na nich).

Przykładowo: zdefiniujemy dziś własny typ `Vector2D`. Obiekty tego typu będą reprezentować wektor na płaszczyźnie. Będą wyposażone w metody, odpowiadające różnym typowym operacjom na wektorach:

```
v1 = Vector2D(1, 0) # tworzenie wektora (1, 0)
v2 = Vector2D(0, 1)
v = 3 * v1 + 4 * v2 # kombinacja liniowa
v.normalize()      # normalizacja wektora
print(v)           # "Vector2D(0.6, 0.8)"
```