# Klasy
# Obiekty

Uniwersytet Ekonomiczny w Poznaniu

1 marca 2016

# Problemy programowania proceduralnego

- Brak możliwości nazywania elementów przy tworzeniu bardziej złożonych struktur danych, np. okrąg, punkt, itd.
- Przy wykorzystaniu krotki (nawet nazwanej) - brak możliwości zmiany elementu składowego.
- Przy wykorzystaniu listy lub słownika - brak możliwości kontrolowania wartości (np. podanie ujemnego promienia okręgu).

# Python, a programowanie obiektowe

- Python pozwala tworzyć programy proceduralne, obiektowe i funkcyjne oraz mieszać te style zależnie od potrzeb.
- Programy proceduralne mogą wykorzystywać obiekty, ale dopiero program tworzący klasy można nazwać obiektowym.

Python jest całkowicie zorientowany obiektowo: można definiować własne klasy, dziedziczyć z własnych lub wbudowanych klas, a także tworzyć instancje zdefiniowanych przez siebie klas.

# Podstawowe pojęcia

- Klasa, typ, typ danych - synonimy, oznaczają zbiór obiektów, ich rodzaj, stanowią "wzorzec" dla tworzenia obiektów w pamięci.

```
s = str('abc')                                    1
type(s)                                           2
```

- Instancja, egzemplarz, obiekt - synonimy, oznaczają pojedyncze wystąpienia, obiekty danej klasy występujące w pamięci.
- Enkapsulacja - obiekty zawierają w sobie dane (**atrybuty**) definiowane przez klasy; klasy definiują funkcje (**metody**), które można wykonywać na obiektach.
- Hermetyzacja - dostęp do atrybutów obiektu powinien być pośredni (przez metody), a nie bezpośredni.
- Właściwości - atrybuty dostępne pozornie bezpośrednio, ale w istocie obsługiwane przez metody.

# Konwencje i reguły dotyczące tworzenia klas

- Nazwy klas w Pythonie zaczynamy dużą literą.
- Metody specjalne `__add__()` lub `__len__()` umieszczone w klasie pozwalają obsługiwać operator + i funkcję `len()` dla obiektów tej klasy.
- Wszystkie metody specjalne zaczynają się i kończą `__`.
- Zwykłe metody zaczynają się od małej litery.
- Można w module zdefiniować dowolną liczbę klas.
- Nazwa modułu nie musi być taka sama jak nazwa klasy.

# Tworzenie własnej klasy

```
class NazwaKlasy:                    1
    blok_instrukcji                  2
```

- `blok_instrukcji` może być pusty - wtedy `pass`.
- Polecenie `class` podobnie jak `def` umożliwia dynamiczne tworzenie klas.
- Metody tworzy się przy użyciu `def`, z tym że do każdej metody trzeba przekazać jawnie instancję obiektu - przez referencję `self`.

# Tworzenie obiektu

- Tworzenie (obiektu) instancji - poprzez wywołanie klasy tak jak funkcji:

```
n = NazwaKlasy()
```

- Obiekty biorą swoje zmienne i funkcje z klas.
- Klasy są podstawowym schematem, według których tworzone są obiekty.

# Zmienne w metodach 1

- Atrybuty obiektu - dostępne po odwołaniu się do instancji `self.x`.
- Zmienne lokalne - dostęp po nazwie.
- Zmienne klasy (statyczne) - dostęp po nazwie klasy.
- Zmienne globalne (modułu) - dostęp po nazwie.

```
modulowa = 'a'                          1
class Testowa:                          2
    z=0                                 3
    def __init__(self, x=1):            4
        self.x=x                        5
    def test(self):                     6
        print(self.x)                   7
        y=7                             8
        print(y)                        9
        print(Testowa.z)                10
        print(modulowa)                 11
```

# Zmienne w metodach 2

- Atrybuty obiektu - dostępne po odwołaniu się do instancji `self.x`.
- Zmienne lokalne - dostęp po nazwie.
- Zmienne klasy (statyczne) - dostęp po nazwie klasy.
- Zmienne globalne (modułu) - dostęp po nazwie.

```
t1 = Testowa()                                    1
t1.test()                                         2
                                                  3
t2 = Testowa(3)                                   4
t2.test()                                         5
```

# Przykład

```
class MojaKlasa:                                                    1
    zmienna=2                                                       2
    def __init__(self,wartosc=1):                                   3
        self.wartosc=wartosc                                        4
    def mojaFunkcja(self):                                          5
        wynik1=self.wartosc**self.zmienna                           6
        wynik2=self.wartosc**MojaKlasa.zmienna                      7
        print(self.wartosc,'podniesiona do potegi:',                8
            self.zmienna,'daje wynik:',wynik1)                      9
        print(self.wartosc,'podniesiona do potegi:',                10
            MojaKlasa.zmienna,'daje wynik:',wynik2)                 11
                                                                    12
liczba1 = MojaKlasa()                                               13
liczba1.mojaFunkcja()                                               14
                                                                    15
liczba1.zmienna = 4                                                 16
liczba1.mojaFunkcja()                                               17
                                                                    18
liczba2 = MojaKlasa(3)                                              19
liczba2.mojaFunkcja()                                               20
```

# Zadanie na rozgrzewkę

- Utworzyć klasę obiektów `Statki`
- Każdy obiekt powinien mieć następujące cechy: model, ładowność, liczba masztów, wielkość.
- Liczba masztów powinna być podana jako liczba (`int`)
- Ładowność powinna być wyliczana z funkcji na podstawie wielkości:
  - jeśli `'duzy'` - ładowność: 50
  - jeśli `'maly'` - ładowność: 10
- Utworzyć trzy nowe obiekty: `statek1`, `statek2` i `statek3`.

# Zadanie 1 - wynagrodzenie

- Obliczyć wynagrodzenie pracowników.
- Obliczyć wysokość składek pracodawcy (na pracownika).
- Obliczyć łączny koszt dla pracodawcy na pracownika.
- Założenia:
  - każdy pracownik powinien być obiektem klasy pracowników,
  - program przyjmuje pierwszą wartość: liczbę pracowników (i),
  - dla liczby pracowników (i) wczytuje imię pracownika i wynagrodzenie brutto (int) (oddzielone spacją),
  - każdy pracownik jest wprowadzany w nowej linii,
  - w wyniku wypisać: imię pracownika, wynagrodzenie netto (float), składki pracodawcy (float), łączny koszt na pracownika (float),
  - każdy pracownik wypisywany jest w nowej linii,
  - w ostatniej linii wypisać łączny koszt dla pracodawcy (wszyscy pracownicy).

# Zadanie 1 - wynagrodzenie

- Przykładowe wejście:

```
2                                    1
Bartek 2580                          2
Marek 1680                           3
```

- Przykładowe wyjście:

```
Bartek 1863.91 535.09 3115.09        1
Marek 1237.20 348.43 2028.43         2
5143.52                              3
```

# Zadanie 1 - podpowiedzi

```
round(i, n) # zaokraglanie

'%.3f' % 1.29003 # daje 1.290
```

- Poszukać sposobu obliczania wynagrodzenia od 01.01.2014
- Za koszty uzyskania przychodu przyjąć zawsze: 111.25

# Zadanie 2 - figury

- Obliczyć sumę pól figur geometrycznych.
- Założenia:
  - program przyjmuje pierwszą wartość: liczbę figur (i),
  - dla liczby figur (i) wczytuje liczby (oddzielone spacją),
  - każda figura jest wprowadzona w nowej linii; jeśli linia zawiera:
    - jedną liczbę - jest to promień koła,
    - dwie liczby - boki prostokąta,
    - trzy liczby - boki trójkąta,
  - w wyniku wypisać sumę pól wprowadzonych figur (zaokrągloną do 2 miejsc po przecinku).
- Dla każdej figury geometrycznej (koło, prostokąt, trójkąt) utwórz oddzielną klasę.

# Zadanie 2 - figury

- Przykładowe wejście:

```
5                                    1
2                                    2
4 6                                  3
3 3.5 4                              4
8 2.1                                5
4.14                                 6
```

- Przykładowe wyjście:

```
112.3                                1
```