Zaawansowane programowanie - lab 2
Kurs po Pythonie II - obiektowość:

Tworzenie klasy:

```
class NazwaKlasy:
      pass
```

Tworzenie obiektu klasy:

```
obiekt = NazwaKlasy()
```

Metody domyślnie zaimplementowane przy tworzeniu klas:

`__init__` - inicjalizacja wartości

```
class NazwaKlasy:
      def __init__(self, atrybut1, atrybut2, ...) -> None:
            self.atrybut1 = atrybut1
            self.atrybut2 = atrybut2
```

`__str__` - reprezentacja tekstowa obiektu

```
class NazwaKlasy:
      def __init__(self, atrybut1, atrybut2, ...) -> None:
            self.atrybut1 = atrybut1
            self.atrybut2 = atrybut2

      def __str__(self):
            return f"Klasa Nazwaklasy z atrybut1: {self.atrybut1} i
atrybut2: {self.atrybut2}"
```

W celu wyświetlenia wszystkich domyślnych metod jak i danych o obiekcie klasy korzysta się z funkcji dir:

```
print(dir(NazwaKlasy(atrybut1, atrybut2))
```

Getter i setter

Getter - metoda, która umożliwia dostęp do wartości atrybuty obiektu
Setter - metoda, która umożliwia ustawienie wartości atrybutu (należy pamiętać o odpowiedniej walidacji)

`@property` tworzy metodę radius, która pozwala na odczytanie wartości atrybutu `_radius` tak, jakby był to zwykły publiczny atrybut.

W celu zdefiniowania atrybutów lub metod prywatnych przed ich nazwą stawiamy znak _

```
class Kolo:
    def __init__(self, promien):
        self._promien = promien

    @property
    def promien(self):
        return self._promien

    @promien.setter
    def promien(self, wartosc):
        if wartosc < 0:
            raise ValueError("Promień nie może być ujemny")
        self._promien = wartosc

kolo = Kolo(5)

print(kolo.promien)

kolo.promien = 10


try:
    kolo.promien = -3
except ValueError as e:
    print(e)

print(kolo.promien)
```

Dziedziczenie

W tym przykładzie klasą bazową jest Zwierze opisującą zwierze domowe, natomiast klasą pochodną jest klasa Pies , która dziedziczy po klasie Zwierze .

```
class Zwierze:
    def __init__(self, imie):
        self._imie = imie

    @property
    def imie(self):
        return self._imie

    @imie.setter
    def imie(self, wartosc):
        if not wartosc:
            raise ValueError("Imię nie może być puste")
        self._imie = wartosc

    def przedstaw_sie(self):
        return f"Jestem {self._imie}"


class Pies(Zwierze):
    def __init__(self, imie, rasa, czy_szczeka_czesto):
        super().__init__(imie)
        self._rasa = rasa
        self._czy_szczeka_czesto = czy_szczeka_czesto

    @property
    def rasa(self):
        return self._rasa

    @rasa.setter
    def rasa(self, wartosc):
        if not wartosc:
            raise ValueError("Rasa nie może być pusta")
        self._rasa = wartosc

    @property
    def czy_szczeka_czesto(self):
        return self._czy_szczeka_czesto

    @czy_szczeka_czesto.setter
    def czy_szczeka_czesto(self, wartosc):
        if not isinstance(wartosc, bool):
            raise ValueError("Wartość dla 'czy szczeka często' musi być typu
boolean (True/False)")
        self._czy_szczeka_czesto = wartosc
```

```
    def przedstaw_sie(self):
        szczeka_info = "szczekam często" if self.czy_szczeka_czesto else "nie
szczekam często"
        return f"Jestem {self.imie}, a moja rasa to {self.rasa}, i
{szczeka_info}"

mój_pies = Pies("Burek", "Labrador", True)
print(mój_pies.przedstaw_sie())
```

Zadania do realizacji:

Zadania proszę przesłać w czasie określonym podczas zajęć w postaci jednego pliku. Rozwiązane zadania należy oznaczyć w kodzie komentarzem `# Zadanie X` . Plik proszę nazwać `lab_2_Imie_Nazwisko.py`

Zadanie 1. Klasa Kot dziedzicząca po klasie Zwierze
Napisz klasę `Kot`, która dziedziczy po klasie Zwierze. Klasa Kot powinna mieć dodatkową właściwość `ulubione_jedzenie` (z getterem i setterem). Dodaj metodę `przedstaw_sie()`, która zwróci imię kota oraz informację o jego ulubionym jedzeniu. Stwórz obiekt tej klasy i wywołaj metodę `przedstaw_sie()` .

Zadanie 2. Klasa Pies z dodatkowymi właściwościami
Rozbuduj klasę `Pies` , dodając dwie nowe właściwości wiek oraz `kolor_siersci` (obie z getterami i setterami). Zapewnij, aby wiek był liczbą całkowitą większą bądź równą zero. Następnie zmodyfikuj metodę `przedstaw_sie()`, aby zwracała pełne informacje o psie, w tym wiek i kolor sierści. Stwórz obiekt klasy `Pies` i przetestuj jego działanie.

Zadanie 3. Klasy według własnego pomysłu
Stwórz dwie klasy, przy czym jedna powinna być bazowa, a druga powinna po niej dziedziczyć. W każdej klasie uwzględnij gettery i settery. Przetestuj ich działanie oraz dodaj krótki komentarz wyjaśniający koncepcję tych klas.