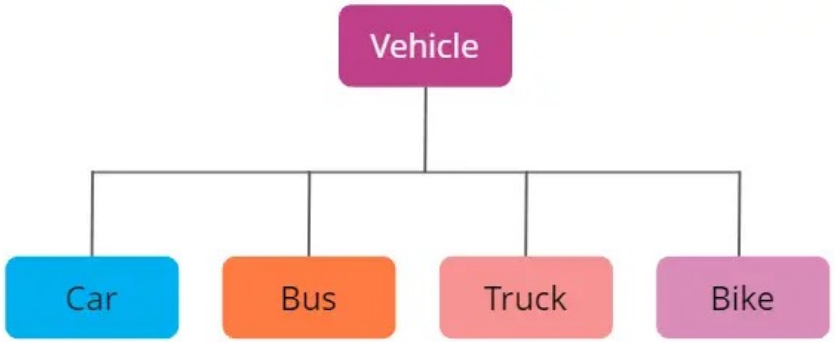
OOP: Dziedziczenie

Podstawy programowania w języku Python



Co to jest dziedziczenie?

- jeden z podstawowych konceptów programowania obiektowego
- polega na tworzeniu nowych klas na bazie istniejących klas
- klasa dziedzicząca może odziedziczyć po klasie bazowej jej pola i metody, a także dodać do nich nowe
- umożliwia tworzenie bardziej złożonych i hierarchicznych struktur klas, co ułatwia organizację i zarządzanie kodem

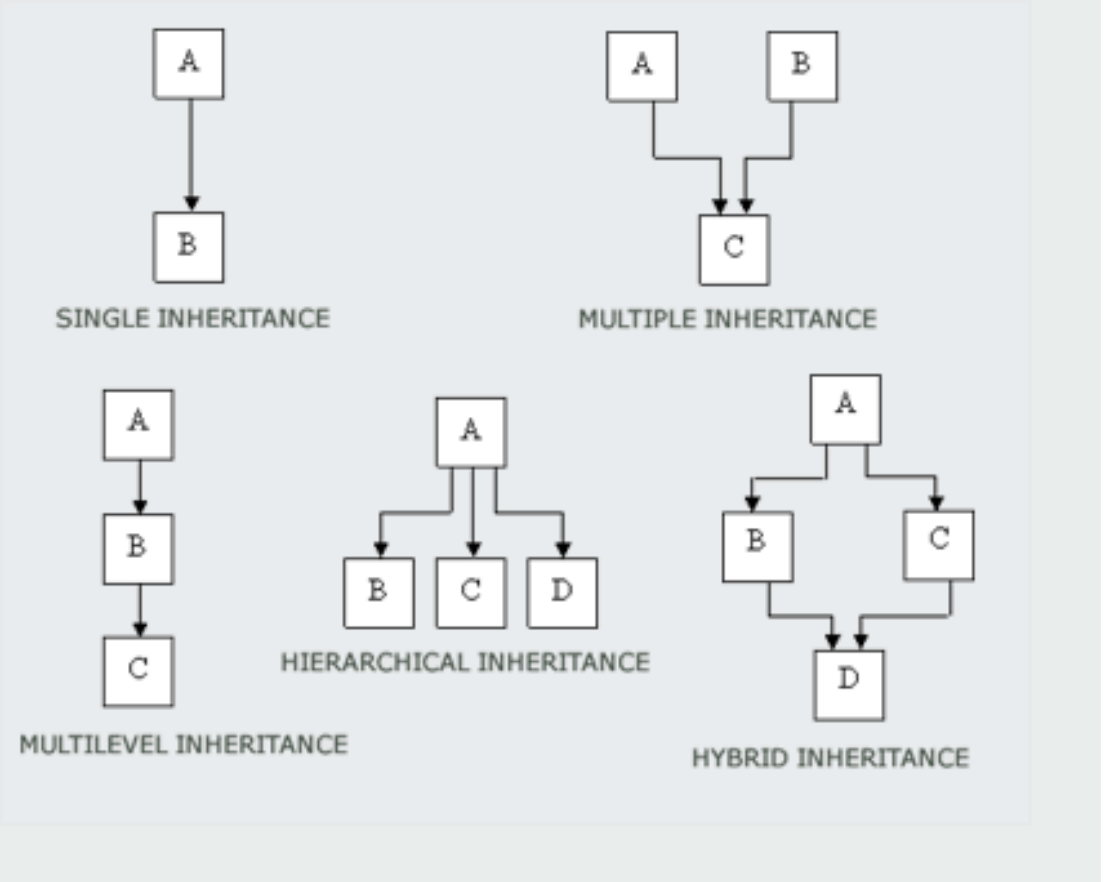
Jak dziedziczyć?

- w deklaracji klasy podajemy nazwę klasy bazowej w nawiasach

```
1  class Animal: # klasa bazowa (nadklasa)
2      pass
3
4  class Cat(Animal): # klasa dziedzicząca (podklasa)
5      pass
```

Rodzaje dziedziczenia

- dziedziczenie może działać na więcej niż jednym poziomie (wielopoziomowe)
- wiele klas może dziedziczyć po tej samej klasie bazowej (hierarchiczne)
- można dziedziczyć po kilku klasach jednocześnie (wielobazowe)



Dziedziczenie wielokrotne (wielobazowe)

- możliwość dziedziczenia po więcej niż jednej klasie bazowej na raz
- w nawiasach okrągłych, umiejscowionych zaraz po nazwie klasy, wstawiamy nazwy klas, z których chcemy dziedziczyć i oddzielamy je przecinkami

```
1   class One:
2       def foo(self):
3           print("Hi!")
4
5   class Two:
6       def bar(self):
7           print("Hello!")
8
9   class MyClass(One, Two):
10      pass
11
12  obj = MyClass()
13  obj.foo() #Hi!
14  obj.bar() #Hello!
```

Weryfikacja związku między klasami oraz obiektami i klasami

- funkcja **issubclass()** potrafi sprawdzić, czy dana klasa jest podklasą jakiejkolwiek innej klasy
- funkcja **isinstance()** weryfikuje czy dany obiekt jest obiektem określonej klasy

```
1  class Animal:
2      pass
3
4  class Cat(Animal):
5      pass
6
7  obj = Cat()
8  print(issubclass(Cat, Animal))
9  print(isinstance(obj, Cat))
```

Polimorfizm

- wiele różnych typów obiektów może być traktowanych w taki sam sposób
- pozwala na wykorzystywanie tych samych metod na wielu różnych obiektach, niezależnie od ich typu
- w Pythonie polimorfizm jest tak naturalny, że nawet się tego nie zauważa

```python
class Animal:
    def speak(self):
        print("???")

class Dog(Animal):
    def speak(self):
        print("Hou hou!")

for animal in [Animal(), Dog()]:
    animal.speak()
```

Zalety dziedziczenia

- umożliwia tworzenie hierarchii klas, w której klasy dziedziczące dziedziczą zachowanie i właściwości z klasy nadrzędnej (unikamy powielania kodu i upraszczamy strukturę programu)
- pozwala na ponowne wykorzystanie kodu z klasy nadrzędnej (jeśli istnieją metody lub właściwości, które są potrzebne w kilku klasach, można je umieścić w klasie nadrzędnej i dziedziczyć je w klasach dziedziczących)
- umożliwia rozszerzanie istniejących klas o nowe metody i właściwości, co pozwala na łatwe rozwijanie i modyfikowanie programu
- wraz z abstrakcją i hermetyzacją umożliwia ukrywanie szczegółów implementacji, co pozwala na tworzenie prostszych i bardziej intuicyjnych interfejsów dla użytkowników

Pytanie

Co to jest dziedziczenie w Pythonie?

a) mechanizm, który pozwala na tworzenie nowych klas na podstawie już istniejących klas
b) mechanizm, który pozwala na tworzenie nowych obiektów na podstawie już istniejących obiektów
c) mechanizm, który pozwala na tworzenie nowych funkcji na podstawie już istniejących funkcji

Odpowiedź: a)

Pytanie

Co wyświetli się na ekranie po wykonaniu skryptu?

```
a) Wrrrrr! Hi!
b) Hi!
c) Wrrrrr!
d) wystąpi wyjątek TypeError
```

```python
class Animal:
    def speak(self):
        print("Wrrrrr!")

class Human:
    def speak(self):
        print("Hi!")

class Hybrid(Human, Animal):
    def speak(self):
        super().speak()

monster = Hybrid()
monster.speak()
```

Odpowiedź: b)

W przypadku dziedziczenia wielobazowego i metod o takich samych nazwach o priorytecie wywoływania metod decyduje kolejność wskazania klas podczas dziedziczenia.

Pytanie

Polimorfizm w Pythonie można opisać jako:

a) mechanizm, który pozwala na łączenie wielu klas w jedną klasę
b) mechanizm, który pozwala na wywoływanie tej samej metody w różnych kontekstach
c) mechanizm, który pozwala na automatyczne zarządzanie pamięcią w Pythonie

Odpowiedź: b)