Abstrakcia

- Abstrakcia alebo abstraktné triedy predstavujú vytvorenie štruktúry správania v abstraktnej triede, ktoré musí byť definované v jej dcérskych triedach
- V abstraktnej triede definujeme správanie, premenné, ktoré musia byť definované v dcérskych triedach
- Dedením z abstraktnej triedy zabezpečíme to, v dcérskej triede musia byť implementované všetky metódy a premenné z abstraktnej triedy
- **Aspoň jedna metóda** v abstraktnej triede musí byť abstraktná, aby trieda bola abstraktná
- Z abstraktnej triedy sa **nevytvárajú** objekty abstraktnej triedy
- Na riadenie abstrakcie existuje v Pythone modul **abc,** z ktorého používame abstraktnú triedu ABC v dedení a dekorátor *abstractmethod* pre abstraktné metódy
- Nevýhoda Pythonu – možnosť vytvárania objektov z abstraktnej triedy, ktorá je prázdna

```python
# Abstraktná trieda
class Stadium(ABC):

    @abstractmethod # abstraktná metóda
    def __init__(self, address):
        pass

    @abstractmethod # abstraktná metóda
    def play_horn(self):
        pass

    def show_info(self):
        print("NHL Game")

    @property
    @abstractmethod # abstraktná premenná, getter
    def address(self):
        pass

    @address.setter # abstraktná premenná, setter
    @abstractmethod
    def address(self, value):
        pass
```

Abstrakcia

- Nevýhoda použitia abstraktného konštruktora v Pythone: v dedenej triede nevieme vynútiť počet parametrov v konštruktore
- Použitím property dekorátoru vieme vytvoriť get() a set() metódy pre inštančnú premennú, ktorá bude **vynútená** v dedenej triede
- Vďaka get() a set() metódam vieme ošetriť vstupy pre takúto premennú, ktorá môže byť nevalidná

** v abstraktnej triede môžeme definovať aj neabstraktnú metódu, ktorú voláme inicializovaným objektom dcérskej triedy

Príklad vráti:

1901 West Madison Street, Chicago

18000

NHL Game

```python
class HockeyStadium(Stadium):
    # Použitím len parametra self s pass argumentom v tele
    # obídeme konštruktor z abstraktnej triedy
    def __init__(self, address, capacity):
        self.address = address
        self.capacity = capacity

    def play_horn(self): # povinná metóda
        pass

    @property
    def address(self):
        return self._address

    @address.setter
    def address(self, value):
        print("Address set")
        self._address = value

    @property
    def capacity(self):
        return self._capacity # povinná premenná

    @capacity.setter
    def capacity(self, value):
        print("Capacity set")
        if value <= 0:
            raise ValueError("Capacity has to be greater than 0")
        self._capacity = value

h_stadium_1 = HockeyStadium("1901 West Madison Street, Chicago", 17000)
print(h_stadium_1.address)

h_stadium_1.capacity = 1800
print(h_stadium_1.capacity)

# Metóda z abstraktnej triedy, ktorá nie je abstraktná
# voláme z objektu zdedenej triedy
h_stadium_1.show_info()
```

Meta triedy

- V Pythone je všetko objekt
- Metatrieda je trieda, ktorá vytvára triedu
- *Type* je základná metatrieda v Pythone
- Trieda *type* vzniká z triedy *type* (platné od Pythonu v3)
- Z metatriedy vytvárame inštancie, ktoré sú triedami našich tried
- Vieme vytvoriť vlastnú metatriedu