3 Objektově orientované programování

Způsob strukturování kódu, kde spojujeme příbuzné metody a proměnné do objektů. V určitých případech je tento přístup přehlednější a praktičtější než jiné (například spojování kusů kódu pomocí logiky nebo funkcí)

- Třída
  - Návrh (blueprint) pro objekt. Většinou obsahuje pouze obecné metody a proměnné
- Objekt
  - Je instance třídy, která již využívá přesně definovaná data. Objek většinou reprezentuje určitý předmět z abstraktního nebo reálného světa
- Metody
  - Funkce devinované v rámci třídy. Samotný objekt je může jen využívat nebo upravit dle svých potřeb
- Vlastnosti
  - Proměnné vytvořené v rámci třídy. Až objekt jim přiřadí hodnotu a nadále s nimi pracuje

```
# vytvoření třídy
class Employee():
  pass

# vytvoření objektu na základě třídy
emp1 = Employee()
```

Proměnné

Python nemá na rozdíl od jiných jazyků přímo implementovanou funkci protected a private proměnných. Místo toho se používají určité konvence

- Protected proměnné
  - Jejich název začíná jedním podtržítkem.
  - Např:

```
_name = "Petr"
```

- Private proměnné
  - Jejich název začíná dvěma podtržítky.
  - Např:

```
__name = "Petr"
```

Konstruktor

Funkce, která se spustí pouze během vytváření objektu. Slouží například k nastavení hodnot proměnných

```
# Vytvoření třídy
class Employee():
  # konstruktor
  def __init__(self, name):
    # slovo "self", zde reprezentuje objekt a je potřeba pro přístup k
vlastnostem daného objektu
    self.__name = name

emp1 = Employee("Petr")
```

Metody

Funkce definované ve třídě.

```
class Employee():
  # konstruktor
  def __init__(self, name):
    # slovo "self", zde reprezentuje objekt a je potřeba pro přístup k
vlastnostem daného objektu
    self.__name = name
  def print_name(self):
    print(self.__name)

emp1 = Employee("Petr")
emp1.print_name()
```

Dědění

Umožňuje vytvořit novou třídu na základě jiné třídy. Nová třída může používat vlastnosti a metody nadřazené třídy. Může je upravovat nebo přepisovat nebo vytvářet vlastní (úpravy provedené v odvozené třídě se do nadřazené třídy nepřenášejí)

```
# Bázová třída
class Human():
  def __init__(self, name, gender):
    self.__name = name
    self.__gender = gender
  def print_info(self):
    print(f"Name: {self.__name}\nGender: {self.__gender}")

class Employee(Human):
  def __init__(self, name, gender, department):
    # slovo "super()" slouží k odkázání na nadřazenou třídu
    super().__init__(name, gender)
    self.__department = department
  def print_info(self):
    super().print_info()
    print(f"Department: {self.__department}")

emp1 = Employee("Petr", "M", "IT")
emp1.print_info()
```

Polymorfizmus

Koncept, který mmá za cíl vytvoření nadřazené obecné třídy, ze které mohou dědit další již zaměřené třídy

```
class Animal():
  def __init__(self, typ, name):
    self.__name = name
    self.__type = typ

  def make_sound(self):
    pass

  def print_info(self):
    print(f"Name: {self.__name}\nType: {self.__type}")

class Cat(Animal):
  def __init__(self, name, sound):
    super().__init__("cat", name)
    self.__sound = sound

  def make_sound(self):
    print(self.__sound)

class Dog(Animal):
  def __init__(self, name, sound):
    super().__init__("dog", name)
    self.__sound = sound

  def make_sound(self):
    print(self.__sound)
    print(self.__sound)

cat1 = Cat("Mačka", "Mňau")
cat1.print_info()
cat1.make_sound()

cat2 = Cat("Gato", "Mňau")
cat2.print_info()
cat2.make_sound()

dog1 = Dog("Hafo", "Haf")
dog1.print_info()
dog1.make_sound()
```

Maturita

- Objekt
- Třída
- Konstruktor
- Dědění
- Ukázka (Zvíře -> kočka/pes)
- Bázová/odvozená třída
- Volání konstruktoru nadřazené třídy
- polymorfizmus

KÓD Z HODINY

```
class Book():
    def __init__(self, name, author, year):
        self.__name = name
        self.__author = author
        self.__year = year
    def info(self):
        return f"Název knihy: {self.__name}\nAutor: {self.__author}\nYear:
{self.__year}"

class Library():
    def __init__(self):
        self.__books = []
    def add_book(self, book):
        self.__books.append(book)
    def print_books(self):
        print("Knihy v systému:")
        for __book in self.__books:
            print(f"{__book.info()}\n---")

lib = Library()
lib.add_book(Book("T", "JK", 1591))
lib.print_books()

class Novel(Book):
    def __init__(self, name, author, year, info):
        super().__init__(name, author, year)
        self.__info = info
        self.__genre = "Novel"
    def info(self):
        return f"{super().info()}\nŽánr: {self.__genre}\nInfo:
{self.__info}"

nov = Novel("G", "LU", "0045", "Pogánek")
lib.add_book(nov)
lib.print_books()
```