**Popište OOP obecně a zdůvodněte proč tento koncept využíváme v programování. Popište základní charakteristiky OOP. Na příkladu v jazyku Python s objekty Geometrick\_utvar\_ -> Ctverec, Obdelnik, Kruh (Viz příloha 2) vysvětlete pojem polymorfismus**

## **Objektově orientované programování (OOP)**

### **Co je OOP a proč ho využíváme?**

OOP (Objektově orientované programování) je programovací paradigma, které umožňuje strukturovat kód pomocí **objektů** a **tříd**. Tento přístup napodobuje reálný svět, kde objekty mají **vlastnosti (atributy)** a **chování (metody)**. Hlavní důvody pro využití OOP:

* **Znovupoužitelnost kódu** – Díky dědičnosti můžeme opětovně využít existující třídy a rozšiřovat je.
* **Lepší organizace kódu** – Kód je přehlednější a lépe škálovatelný.
* **Zapouzdření** – Skrýváme detaily implementace a zpřístupňujeme jen potřebné části.
* **Polymorfismus** – Umožňuje používat stejné rozhraní pro různé datové typy.

### **Základní charakteristiky OOP**

1. **Třídy a objekty** – Třída je šablona, podle které se vytvářejí objekty.
2. **Dědičnost** – Umožňuje, aby jedna třída přebírala vlastnosti a metody jiné třídy.
3. **Zapouzdření** – Ochrana dat a metod uvnitř objektu.
4. **Polymorfismus** – Možnost volat stejnou metodu na různých třídách.

## **Polymorfismus na příkladu v Pythonu**

Polymorfismus znamená, že stejná metoda může mít různé chování v různých třídách. V poskytnutém kódu je polymorfismus vidět v metodách vypocti\_obvod(), vypocti\_obsah() a params(), které jsou definovány v základní třídě Geometricky\_utvar, ale každá dědičná třída (Ctverec, Obdelnik, Kruh) si je implementuje jinak.

class Geometricky\_utvar:  
 def vypocti\_obvod(self):  
 return None  
 def vypocti\_obsah(self):  
 return None

Třída Geometricky\_utvar definuje metody, ale nemá jejich konkrétní implementaci.

class Ctverec(Geometricky\_utvar):  
 def vypocti\_obvod(self):  
 return 4 \* self.strana  
 def vypocti\_obsah(self):  
 return self.strana \*\* 2

V třídě Ctverec jsou tyto metody implementovány konkrétně pro čtverec. To samé platí pro Obdelnik a Kruh.

Díky polymorfismu můžeme stejným způsobem pracovat s různými tvary:

utvary = [Ctverec(5), Kruh(4), Obdelnik(5,2)]  
for u in utvary:  
 print(u.nazev, 's parametry', u.params(), 'ma obvod', u.vypocti\_obvod(), 'a obsah', u.vypocti\_obsah(), '.')

I když voláme vypocti\_obvod() a vypocti\_obsah() na objektech různého typu, Python automaticky spustí odpovídající implementaci pro každý konkrétní objekt. **To je polymorfismus v praxi!**