**OOP enkapsulace, Instalace aplikačního SW v Unix-like systémech, GPIO Raspberry Pi 4**

**I. OOP – Objektově orientované programování a enkapsulace**

1. **Co je to OOP?**

* **Objektově orientované programování (OOP)** je **styl (paradigma) programování**, kde místo psaní dlouhých sekvencí příkazů vytváříme **objekty**.
* Objekt je jako **věc** ze skutečného světa – třeba **auto**, **člověk**, **pes**.
* Každý objekt má:
  + **Atributy (vlastnosti)** – např. člověk má "jméno", "věk".
  + **Metody (činnosti, funkce)** – např. člověk "mluví()", "chodí()".

2. **Proč používáme OOP?**

* **Zjednodušení a přehlednost kódu**.
* **Znovupoužitelnost** – jednou vytvořím "Auto" a pak vyrábím různé modely (objekty).
* **Bezpečnost dat** – díky **enkapsulaci** (viz níže).
* **Snadná údržba a rozšíření** – přidám novou funkci bez přepisování všeho.

3. **Základní principy OOP:**

Obsah obrázku text, snímek obrazovky, Písmo, účtenka

Obsah vygenerovaný umělou inteligencí může být nesprávný.4. **Co je to enkapsulace? (hlavní bod)**

* **Schování vnitřních údajů (atributů) objektu před okolím**.
* Když mám objekt "Auto", tak **motor** a jeho stav chci **schovat** a přistupovat k němu **pouze přes metody**.
* **Chrání data před nechtěnou změnou**.

5. **Příklad enkapsulace v Pythonu na třídě Geometricky\_utvar:**

* Obsah obrázku text, Písmo, snímek obrazovky, algebra

  Obsah vygenerovaný umělou inteligencí může být nesprávný.\_\_nazev je **soukromý atribut**, ke kterému **nejde přistupovat přímo** (např. obj.\_\_nazev nejde!).
* Používám metody get\_nazev() a set\_nazev(), aby **kontrolovaně** pracoval s hodnotou.
* ✅ **Výhoda:**
* Když např. zadám špatný název, metoda může zkontrolovat vstup