**26.**

**OOP: rozdíly oproti procedurálnímu paradigmatu, návrh objektů, zapouzdření, polymorfismus**

**Procedurální paradigma**

Vytváření programů pomocí funkcí nebo procedur, které manipulují daty.

**Příklad:**

**def** soucet(a, b):

**return** a + b

**def** rozdil(a, b):

**return** a - b

**# Hlavní program**

cislo1 = **int**(input("Zadej první číslo: "))

cislo2 = **int**(input("Zadej druhé číslo: "))

vysledek\_soucet = soucet(cislo1, cislo2)

print("Součet:", vysledek\_soucet)

vysledek\_rozdil = rozdil(cislo1, cislo2)

print("Rozdíl:", vysledek\_rozdil)

**Objektově orientované paradigma**

Zaměřuje se na modelování entit pomocí objektů, které kombinují data a funkce (metody) pracující s těmito daty.

**Příklad:**

**class** Kalkulacka:

**def** \_\_init\_\_(**self**, cislo1, cislo2):

self.cislo1 = cislo1

self.cislo2 = cislo2

**def** soucet(**self**):

return self.cislo1 + self.cislo2

**def** rozdil(**self**):

return self.cislo1 - self.cislo2

**# Hlavní program**

cislo1 = **int**(input("Zadej první číslo: "))

cislo2 = **int**(input("Zadej druhé číslo: "))

kalkulacka = Kalkulacka(cislo1, cislo2)

print("Součet:", kalkulacka.soucet())

print("Rozdíl:", kalkulacka.rozdil())

**Návrh objektů**

V OOP je program strukturován kolem objektů, které jsou instancemi tříd. Každý objekt může mít své vlastní **atributy** (data) a **metody** (funkce), které s těmito daty pracují.

**Zapouzdření**

Skrytí vnitřku objektu a poskytnutí rozhraní (metod) pro komunikaci s objektem.

To znamená, že vnější svět nemá přímý přístup k datům objektu, ale musí komunikovat s ním pomocí jeho metod.

**Příklad:**

**class** BankovyUcet:

**def** \_\_init\_\_(**self**, jmeno, saldo=0):

**self**.\_\_jmeno = jmeno # Privátní atribut

**self**.\_\_saldo = saldo # Privátní atribut

**def** get\_jmeno(**self**):

return **self**.\_\_jmeno

ucet = BankovyUcet(“Jan Novak“)

print(ucet.get\_jmeno())

**Polymorfismus**

Stejná metoda nebo funkce může být použita pro různé typy objektů a vykonávat se různé akce podle typu objektu, s nímž se pracuje.

**Příklad:**

**class** Zvire:

**def** zvuk(**self**):

**pass**

**class** Pes(**Zvire**):

**def** zvuk(**self**):

**return** "Haf!"

**class** Kocka(Zvire):

**def** zvuk(**self**):

**return** "Mnau!"

**# Použití polymorfismu**

zvir1 = Pes()

zvir2 = Kocka()

print(zvir1.zvuk()) # Vypíše: Haf!

print(zvir2.zvuk()) # Vypíše: Mnau!

Zde funkce **zvuk()** je polymorfní, protože její chování se mění podle toho, zda je volána na instanci třídy Pes nebo Kocka. To umožňuje použití stejného kódu pro různé typy zvířat