Metodický list

Č.3

Názov témy: Objektové programovanie	
Cieľová skupina:	- 4. ročník strednej školy
Predmet:	- Informatika
Ciele:	- vzdelávací cieľ: oboznámiť sa s objektovým programovaním, vedieť samostatne vytvárať triedy a objekty a rozumieť princípu tohto druhu programovania - výchovný cieľ: viesť k samostatnosti a k použitiu vedomostí v praxi - cieľ je zameraný na zvyšovanie informačnej gramotnosti a programovacích schodností
Organizačné formy:	- **O**svojovanie nových vedomostí, využívanie vedomostí a zručností v praxi - Práca žiakov – individuálna, frontálna
Organizačné metódy:	● vysvetľovanie, názorná ukážka
Popis:	▪ Úvodná motivácia Rozdiel medzi štrukturovaným programovaním a objektovým programovaním. ▪ Vysvetlenie učiva a názorná ukážka ▪ Čo je to objekt, trieda, vlastné triedy, atribúty, SELF a __init__
Zadanie pre žiakov:	Žiaci si zapisujú výklad učiteľa, odpovedajú na otázky
Príprava, učebné pomôcky:	Dataprojektor

Metodický postup:	• Úvod hodiny Učiteľ v úvode žiakom vysvetlí v čom spočíva základný rozdiel medzi - u nich - doteraz naučením spôsobom programovania a novým objektovým spôsobom. Pripraví žiakov na náročný výklad, poprosí žiakov aby si zapisovali zložitejšie poučky a definície. Vzhľadom na náročnosť povolí žiakom pri akejkoľvek nezrovnalosti prerušiť výklad učiteľa prihlásením a položením otázky. • Výklad učiva a názorná ukážka **Hodnoty a objekty** Predtým než sa učiteľ zameria na triedy, začne objektami. Vysvetlenie čo to vlastne je objekt v pythone: Každá hodnota — to značí niečo čo môžeme uložiť do premennej, vrátiť z funkcie alebo napr. zoznamu — je objekt. Existujú jazyky kde existujú aj iné hodnoty než objekty (JavaScript, C++, JAVA..), existujú tiež jazyky kde objekty vôbec niesu (C). Ale v Pythone medzi hodnotou a objektom nie je rozdiel, takže je na jednu stranu trochu zložitejšie v čom spočíva ta „objektovosť“, ale na druhú stranu to nie je potrebné vedieť do detailov. Základnou vlastnosťou objektov je to, že obsahujú dáta(informácie) a aj „chovanie“ inštrukcie alebo metódy ktoré s týmito dátami pracujú. Napr. reťazce v Pythone obsahujú informácie (sekvenciu znakov) a aj užitočné metódy ako UPPER alebo COUNT. Keby reťazce neboli objekty, musel by Python mať veľké množstvo funkcií ako str_upper a str_count. **Objekty teda spojujú dáta a funkčnosť dohromady.** Učiteľ položí žiakom otázku: Myslite si že LEN je funkcia alebo metóda objektu? Ak žiaci odpovedia, že ide o funkcie majú pravdu, učiteľ teda dodá že nie všetko v Pythone je na 100% objektové. **Triedy** Data každého objektu sú špecifické pre konkrétny objekt („abc“, obsahuje iné znaky než „def“), ale funkčnosť — metódy — bývajú spoločné pre všetky objekty daného typu. Trebárs metóda COUNT() by sa dala napísať zhruba takto: Učiteľ začne v editore programovať: `def count(retazec, znak):` `  pocet = 0` `  for c in retazec:` `  if c == znak:` `  pocet = pocet + 1` `  return pocet` Zdrojový kód žiakom poslúži ako príklad, v ktorom je vidieť že aj keď je funkcia vracia za každým inú hodnotu pre každý reťazec, samotná metóda je spoločná všetkým reťazcom. Toto spoločné chovanie určuje typ (ang. Type) alebo **trieda** (ang. Class) daného objektu. Učiteľ však dodá že v histórií jazyka Python bol rozdiel medzi „typom“ a „triedou“ ale dneska sa už

jedná o synonyma.
Typ objektu vie zistiť funkcia type:

```
>>> type(0)
<class 'int'>
>>> type(True)
<class 'bool'>
>>> type("abc")
<class 'str'>
>>> with open('subor.txt') as f:
...     type(f)
...
<class '_io.TextIOWrapper'>
```

Zdrojový kód poukazuje že TYPE vracia nejaké triedy. Následne čoho sa učiteľ spýta žiakov či už tušia, čo to vlastne trieda je. Ide vlastne o popis, ako sa chovajú všetky objekty daného typu. Ale dajú sa, podobne ako funkcie zavolať. Trieda teda väčšinou obsahuje nielen popis ako sa objekty daného typu budú chovať, ale aj návod ako také objekty vytvoriť.

Učiteľ sa spýta žiakov: Myslíte že STR, INT a FLOAT je funkcia alebo trieda? Ide samozrejme o triedu, čo učiteľ potvrdí programovaním:

```
>>> str
<class 'str'>
>>> type('abcdefgh')
<class 'str'>
>>> type('abcdefgh') == str
True
```

Vlastné triedy

V ďalšej fáze učiteľ vysvetlí prečo je dobré naprogramovať si vlastnú triedu. Je dobré naprogramovať vlastnú triedu keď potrebujeme mať vo svojom programe viacero objektov s podobným chovaním. Ku príkladu učiteľ spomenie že kartová hra by mohla mať triedu karta, webová aplikácia triedu užívateľ, tabuľkový procesor triedu riadok… Otázka na žiakov: Viete povedať podobný príklad?
Ku príkladu učiteľ naprogramuje:

```
class Maciatko:
       def zaMnaukaj(self):
        print("Mňau!")
```

Tak ako sa funkcie definujú pomocou DEF, triedy majú kľú%cové slovo CLASS, za torú napíseme meno triedy, dvojbodku, a potom

odsadené telo triedy. Učiteľ tiež spomenie fakt že triedy sa pomenuvavajú zasadne s velkým zaciatocnim pismenom. Učiteľ znova pokračuje programovaním:

```
# Vytvorenie konkretneho objektu
maciatko = Maciatko()
# Volanie metody
maciatko.zaMnaukaj()
```

Atribúty

Učiteľ definuje atribút ako:

Objekty vytvorené z vlastných tried ktoré majú jednu vlastnosť, akú triedy ako STR nedovoľujú, možnosť si definovať vlastné atribúty — **Informácie ktoré sa uložia k danému objektu**

```
micka = Maciatko()
micka.meno = 'Micka'
```

SELF

Učiteľ podotkne na skutočnosť ze sa na chvíľku vrátia k metódam, konkrétne k parametru SELF. Pričom každá metóda ma prístup ku konkrétnemu objektu, na ktorom pracuje, práve cez argument SELF. Učiteľ znova pre ukážku programuje do editora, použitie argumentu SELF:

```
class Maciatko:
        def zaMnaukaj(self):
        print("{}: Mňau!".format(self.meno))
micka = Maciatko()
micka.meno = 'Micka'
micka.zaMnaukaj()
```

Načo učiteľ poukáže na skutočnosť že výraz *micka.zaMnaukaj* urobí metódu, ktorú, keď zavoláme, predá objekt *micka* ako prvý argument funkcie *zaMnaukaj*.

Učiteľ položí otázku na žiakov čim sa teda líši metóda od funkcie? Žiaci odpovedajú že tým že metóda si pamätá objekt na ktorom pracuje.

__init__

Učiteľ poukáže aj na ďalšie možnosti posielania argumentov metódam. Ide práve o špeciálnu metódu __init__ ktorú využívame napr. pokiaľ chceme nastaviť *macicke* meno už pri vytvorení objektu.

Micka= Maciatko(meno='Micka'

Metóda __init__ sa zavolá automaticky, keď sa vytvorí novy objekt

```
class Maciatko:
    def __init__(self, meno):
```

	`` self.meno = meno def zaMnaukaj(self): print("{}: Mňau!".format(self.meno)) Micka = Maciatko('Micka') Micka.zaMnaukaj() ``` Po čom učiteľ vysvetlí že už nie je možnosť, ako vytvoriť Mačiatko bez mena, takže *zaMnaukaj b*ude vždycky fungovať. • Záver Učiteľ žiakom poďakuje za pozornosť a za aktivitu pri zodpovedaní otázok. Pripomenie o čom bude nasledujúca dvojhodinovka.
Hodnotenie: (spätná väzba)	- Učiteľ ústne zhodnotí prácu žiakov na základe zodpovedaných otázok
Postrehy z overovania:	
Časová dotácia:	- 2 vyučovacie hodiny
Prílohy:	- Prezentácia s výkladom učiva, zoznam internetových zdrojov k rozšíreniu či zopakovaniu znalostí o učive