# **16. Soubory a serializace – Ukládání a načítání dat, formáty souborů**

# **Soubory**

**Co jsou to soubory?**

**Uložiště dat s různými datovými typy (text, obrázky, zvuk, videa)**

**Umožňuje přístup k datům**

V práci se soubory je **vždy potřeba být opatrný a nikdy nespoléhat na „existenci“ a „správnost souboru“**

Soubory jsou v programování také způsob, jak si uložit data na dobu delší, než je pouze běh programu

* **Je dobrým zvykem používat v každé práci se soubory výjimky**
* Soubor nemusí existovat / může být poškozený

**Dělí se (vždy binární)**

* Binární
* pro člověka **nečitelné soubory**
* Jsou psány kombinací NUL a JEDNIČEK
* Způsob ukládání informací do paměti úsporněji
* Pro otevření binárního souboru programem, je potřeba vědet jakým způsobem je kódován
* Textové
* Text
* Lidsky čitelné
* Executable
* Spustitelné soubory

**Práce se soubory PYTHON**

******Potřebujeme znát umístění souboru a k tomu lze použít ***importovaný modul OS***

* ******Získat ***cestu ve který se nachází***
* ******Získat ***všechny soubory na daném adresáři***
* ******Zjistit, *zda* ***soubor opravdu existuje (true/false)***
* Pro otevření souboru použijeme BUILD-IN metodu OPEN() do který napíšeme adresu souboru
  + Soubor následně uložíme do proměnné a můžeme s ním operovat
  + Základní operace:
    - Read(), readline()
    - Write() – pokud soubor už existuje, tak ho přepíše !!!
      * Append() – přidá do souboru řádek (nepřepíše)
    - Close() – zavření souboru
    - Os.rename() – přejmenování
    - Os.remove() – smazání
    - Os.path.getsize() – velikost souboru
    - Truncate() – maže obsah souboru aniž by smazalo samotný soubor

# Formáty souboru

# **Serializace**

Velká část instancí je používaná krátce a poté smazána z paměti garbage collectorem.

Některé objekty/instance občas potřebujeme uložit na delší dobu.

Toto označujeme jako ***Serializaci***.

* Proces convertování objektu nebo dat do formátu který může být uložen nebo přenesen
* Typicky STRING OF BYTES

Pokud ona data potřebujeme zpětně přečíst, provedeme tzv. ***Deserializaci*** (vytvoříme instanci)

* ***Serializace*** je proces transformující objekt v operační paměti na stream bytů
* ***Deserializace*** je opačný proces – rekonstruující objekt do operační paměti ze streamu bytů do stejného stavu, ve kterém byl objekt serializován
* ***Použití serializace:***
  + Serializace je užitečná pro síťové aplikace (aplikace v jednom uzlu může objekt serializovat a aplikace v jiném uzlu ho deserializuje)
  + Ukládání nastavení aplikace atd...
* Serializovat lze různými způsoby

*Hůře lidsky čitelné:*

* + - ***Binárně***

*Lépe lidsky čitelné:*

* + - ***JSON***
    - ***XML***

Serializace v PYTHON

Použití importu PICKLE

A picture containing text, screenshot, font

Description automatically generated



Deserializace v PYTHON





***Binárně***

* Přes třídu ***BinaryFormatter***, která se vytvoří jako objekt s prázdným konstruktorem
* Serializuje se přes metodu ***Serialize***, do které vstupuje *stream* a *objekt*
* Deserializuje se přes metodu ***Deserialize***, do které vstupuje *stream*
* Výstupem je binární soubor, který může mít ale jakoukoliv příponu

***JSON***

* Interpretuje objekty jako text s notací jako JavaScript
* Základní pravidla pro JSON syntax
  + ***Hodnoty*** jsou ve formátu ***název:hodnota (“jmeno”:”Pavel”)***
  + Jednotlivá ***data*** jsou oddělena ***čárkami ,***
  + ***Objekty*** jsou označeny ***složenými závorkami {}***
  + ***Pole*** jsou označena ***hranatými závorkami []***
* Používá se statická třída ***JsonSerializer***
* Pro serializaci se používá metoda ***json.dumps(data)***

A picture containing text, screenshot, font

Description automatically generated

* Pro deserializaci se používá metoda ***json.loads(serialized\_json)***

A picture containing text, font, screenshot

Description automatically generated

***XML***

* Interpretuje objekty ve stylu HTML tagů
* Je potřeba vytvořit instanci třídy ***XmlSerializer***, která vyžaduje do konstruktoru ***typ***, který chceme serializovat/deserializovat (***typeof***)
* Serializace se provádí pomocí metody ***Serialize***, která vyžaduje *stream* a *objekt*
* ***Třída, kterou serializujeme musí splňovat***
  + Je public
  + Obsahuje prázdný konstruktor
  + Vlastnosti, které chceme serializovat jsou public, jinak se vynechají
* Jak bude výsledný soubor vypadat si jde upravit v programu pomocí atributů
  + Například **[XmlRoot(“Nazev”)]**, změní název kořene XML stromu