# Soubory a serializace - Ukládání a načítání dat, formáty souborů

## Ukládání a Načítání dat

Soubory jsou způsob, jak si uložit data na dobu delší, než je pouze běh programu. Práce se soubory by měla být ohlídána na výjimky, jelikož soubory se můžou poškodit nebo ani nemusí existovat. Například pokud soubor neexistuje a chci do něj zapisovat dostanu chybovou hlášku FileNotFoundError.

**open**(*file*, *mode='r'*, *buffering=- 1*, *encoding=None*, *errors=None*, *newline=None*, *closefd=True*, *opener=None*)

| **Character** | **Meaning** |
| --- | --- |
| 'r' | open for reading (default) |
| 'w' | open for writing, truncating the file first |
| 'x' | open for exclusive creation, failing if the file already exists |
| 'a' | open for writing, appending to the end of file if it exists |
| 'b' | binary mode |
| 't' | text mode (default) |
| '+' | open for updating (reading and writing) |

* buffering (optional) - used for setting buffering policy
* encoding (optional) - the encoding format
* errors (optional) - string specifying how to handle encoding/decoding errors
* newline​ (optional) - how newlines mode works (available values: None, ' ', '\n', 'r', and '\r\n'
* closefd (optional) - must be True (default); if given otherwise, an exception will be raised
* opener (optional) - a custom opener; must return an open file descriptor

with open('spamspam.txt', 'w') as f:

f.write(“Ahoooj”)

with open('spamspam.txt', 'r') as f:

in\_file = f.read()

## Serializace a Deserializace

Velká část instancí je používaná krátce a poté smazána z paměti garbage collectorem. Některé objekty/instance občas potřebujeme uložit na delší dobu, například do perzistentního úložiště (Relační DB). Toto označujeme jako Serializaci. Pokud ona data potřebujeme zpětně přečíst, provedeme tzv. Deserializaci

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

Pickle modul se používá pro serializaci a deserializaci struktur v python. Jakýkoliv objekt v python může být serializován aby následně mohl být uložen na disk. Pickle serializuje objekt a následně převede do řetězce charakterů aby tento řetězec obsahující všechny informace ohledně objektu mohl být deserializován znovu třeba v jiném python program.

### Pickle dump:

import pickle

# take user input to take the amount of data

number\_of\_data = int(input('Enter the number of data : '))

data = []

# take input of the data

for i in range(number\_of\_data):

raw = input('Enter data '+str(i)+' : ')

data.append(raw)

# open a file, where you ant to store the data

file = open('important', 'wb')

# dump information to that file

pickle.dump(data, file)

# close the file

file.close()

### Pickle load:

import pickle

# open a file, where you stored the pickled data

file = open('important', 'rb')

# dump information to that file

data = pickle.load(file)

# close the file

file.close()

print('Showing the pickled data:')

cnt = 0

for item in data:

print('The data ', cnt, ' is : ', item)

cnt += 1

## Formáty souborů

### JSON

import json

data = {"name": "John", "age": 30, "city": "New York"}

# Serialize data to JSON

json\_data = json.dumps(data)

# Save data to a file

with open("data.json", "w") as file:

file.write(json\_data)

# Load data from a file

with open("data.json", "r") as file:

data = json.load(file)

print(data)

### CSV

import csv

data = [

{"name": "John", "age": 30, "city": "New York"},

{"name": "Jane", "age": 25, "city": "Los Angeles"},

{"name": "Bob", "age": 40, "city": "Chicago"}

]

# Save data to a CSV file

with open("data.csv", "w", newline="") as file:

writer = csv.DictWriter(file, fieldnames=["name", "age", "city"])

writer.writeheader()

for item in data:

writer.writerow(item)

# Load data from a file

with open("data.csv", "r", newline="") as file:

reader = csv.DictReader(file)

print(reader.fieldnames)

for item in reader:

print(item)

### XML

import xml.etree.ElementTree as ET

data = [{"name": "John", "age": 30, "city": "New York"},

{"name": "Pepa", "age": 40, "city": "York"}]

# Create XML element

root = ET.Element("person")

for value in data:

for key, value in value.items():

ET.SubElement(root, key).text = str(value)

# Save data to an XML file

tree = ET.ElementTree(root)

tree.write("data.xml")

# Load data from a file

data = ET.parse('./data.xml')

for value in data.iter("person"):

for element in value:

print(element.tag, element.text)