ΠΕΡΙΕΧΟΜΕΝΑ:

1. Βασικά Στοιχεία (Υπενθυμίσεις από μάθημα 15)
2. Παραμετροποίηση dump[s]
3. Σειριοποίηση
4. Αποσειριοποίηση
5. … και δύο τεχνικές

Αθανάσιος Σ.

Χρυσός Χορηγός Μαθήματος

Χάρης Κικίδης

Χρυσός Χορηγός Μαθήματος

Υπενθυμίσεις από Python - Μάθημα 15 (Αρχεία):

- Οι ακόλουθες μέθοδοι υποστηρίζουν την αποθήκευση/ ανάκτηση δεδομένων σε/από αρχείο στη μορφή JSON

Μέθοδος	Επεξήγηση
dump(obj, file)	Αποθηκεύει στο αρχείο file το αντικείμενο obj σε μορφή JSON
load(file)	Φορτώνει από το αρχείο το JSON, το μετατρέπει σε αντικείμενο της Python και το επιστρέφει

- Οι ακόλουθες μέθοδοι υποστηρίζουν την μετατροπή δεδομένων σε συμβολοσειρά στη μορφή JSON (και αντίστροφα)

Μέθοδος	Επεξήγηση
dumps(obj)	Μετατρέπει σε συμβ/ρα μορφής JSON το python αντικείμενο obj και την επιστρέφει
loads(str)	Μετατρέπει τη συμβ/ρά που απεικονίζει αντικ/νο JSON σε python αντικ/νο και το επιστρέφει

  - **int, float, boolean, list, dictionary** μετατρέπονται κανονικά
  - **tuple**: μετατρέπεται σε πίνακα (~list) του JSON
  - **set**: δεν είναι δυνατή η μετατροπή του σε JSON (δες και json_datatypes.py)

- Η μετατροπή αντικειμένου σε JSON λέγεται και σειριοποίηση (**serialization**)
- Η μετατροπή JSON σε αντικείμενο λέγεται και αποσειριοποίηση (**deserialization**)

Παράδειγμα 1: json_basics/json_basics.py

```
import json
from random import randrange

with open("ob.json", "w") as f:
    json.dump([(1,2),(2,3)], f)
with open("ob.json", "r") as f:
    ob = json.load(f)
    print(ob)

with open("multi_line.json", "w") as f:
    for _ in range(4):
        v = randrange(3)
        if v == 0:
            json.dump([1,2,3], f)
        elif v == 1:
            json.dump({"a":1, "b":2, "c":3}, f)
        else:
            json.dump(4, f)
        f.write("\n")

with open("multi_line.json", "r") as f:
    for lineObj in f:
        print(json.loads(lineObj))
```

2. Παραμετροποίηση dump[s]

- Μπορούμε να παραμετροποιήσουμε περαιτέρω τη dump() και τη dumps() με τα εξής ορίσματα με λέξεις κλειδιά:

Όρισμα και default τιμή	Επεξήγηση
skipKeys=False	Αν τεθεί True, τότε για τα κλειδιά λεξικών: Αν το κλειδί δεν είναι κάποιος βασικός τύπος (str, int, float, bool, None), τότε θα παρακαμφθούν αντί να προκαλέσουν εξαίρεση ValueError [Σημείωση: Τα JSON αντικείμενα έχουν ως κλειδιά αποκλειστικά συμβολοσειρές. Οι τιμές μπορεί να είναι string, number, object, array, boolean, null]
indent=None	Αν τεθεί ίσο με N, τότε γίνεται στοίχιση N χαρακτήρων στα επίπεδα στοίχισης των αντικειμένων του αρχείου
separators	Αν indent = None: default τιμή: (", ", ":") Αλλιώς default τιμή: (", ", ": ") Για ελαχ/ση μεγέθους αρχείου: (",", ":")
sort_keys=False	Αν sort_keys=True, τότε τα κλειδιά του αντικειμένου εμφανίζονται ταξινομημένα.
default=None	Τίθεται ίσο με μία συνάρτηση που κωδικοποιεί σε JSON ένα αντικείμενο κλάσης (επομ. διαφάνεια)

- Υπάρχουν και άλλα, πιο προχωρημένα χαρακτηριστικά που καθορίζονται με ορίσματα με λέξεις - κλειδιά.

Παράδειγμα 2: json_args/beautify.py

```
import json
a_person = {
   "first_name": "John",
   "last_name": "Doe",
   "age": 25
}
with open("ob.json", "w") as f:
   json.dump(a_person, f, indent=2, sort_keys=True)
```

Η load() και η loads() μπορούν να παραμετροποιηθούν πρόσθετα με τα εξής ορίσματα με λέξεις - κλειδιά:

Όρισμα και default τιμή	Επεξήγηση
parse_float=None	Συνάρτηση που καλείται όταν διαβαστεί πραγματικός
parse_int=None	Συνάρτηση που καλείται όταν διαβαστεί ακέραιος

Παράδειγμα 3: json_args/json_dumb_args.py

```
with open("ob.json", "r") as f:
   x = json.load(f, parse_int=decimal.Decimal,
                 parse_float=decimal.Decimal)
```

Σειριοποίηση (serialization):

- Η μετατροπή ενός αντικειμένου κλάσης σε μορφή που να είναι αποθηκεύσιμη σε αρχείο

Α' τρόπος σειριοποίησης:

- Κατασκευάζουμε μία συνάρτηση που επιστρέφει τα μέλη του αντικειμένου κωδικοποιημένα ως JSON δεδομένα (ενδ. λεξικό)
- Καλείται η dump[s] με όρισμα στην παράμετρο με λέξη-κλειδί **default** το όνομα της συνάρτησης.

Παράδειγμα 4: json_objects_manip_func

```
class Time:
    def __init__(self, hour, minute, second):
        self.hour = hour
        self.minute = minute
        self.second = second
```

```
def time_encoder(time):
    if isinstance(time, Time):
        return {"hour": time.hour, "minute": time.minute,
                "second": time.second}
    else:
        raise TypeError("Type should be Time")
```

```
with open("ob.json", "w") as f:
    json.dump(t1, f, default=time_encoder)
```

Β' τρόπος σειριοποίησης:

- Κατασκευάζουμε μία κλάση που κληρονομεί την json.JSONEncoder
  - Στην οποία επαναορίζουμε την μέθοδο default(self, ob) η οποία επιστρέφει το σειριοποιημένο αντικείμενο
- Καλείται η dump[s] με όρισμα στην παράμετρο με λέξη-κλειδί **cls** το όνομα της παραπάνω κλάσης.

Παράδειγμα 5: json_objects_manip_class

```
class Time:
    ...
```

```
class TimeEncoder(json.JSONEncoder):
    def default(self, time):
        if isinstance(time, Time):
            return {"hour": time.hour, "minute": time.minute,
                    "second": time.second}
        else:
            raise TypeError("Type should be Time")
```

```
with open("ob.json", "w") as f:
    json.dump(t, f, cls=TimeEncoder)
```

- Αναμένεται TypeError εξαίρεση για αντικείμενο λάθους τύπου.
- Η κλάση json.JSONEncoder δίνει πλούσια λειτουργικότητα με επιπλέον μεθόδους.

Αποσειριοποίηση (serialization):

- Η μετατροπή αποθηκευμένης πληροφορίας σε αρχείο JSON, σε μορφή επεξεργάσιμη από τη γλώσσα (εδώ json=>αντικείμενο)

Α' τρόπος αποσειριοποίησης:

- Κατασκευάζουμε μία συνάρτηση η οποία δέχεται ένα λεξικό και επιστρέφει ένα αντικείμενο από τις τιμές του λεξικού
- Καλείται η load[s] με όρισμα στην παράμετρο με λέξη-κλειδί **object_hook** το όνομα της συνάρτησης.

Παράδειγμα 6: json_objects_manip_func

```
class Time:
    def __init__(self, hour, minute, second):
        self.hour = hour
        self.minute = minute
        self.second = second
```

```
def time_decoder(time):
    return Time(time["hour"], time["minute"], time["second"])
```

```
with open("ob.json", "r") as f:
    t2 = json.load(f, object_hook=time_decoder)
    print(t2)
```

Β' τρόπος σειριοποίησης:

- Κατασκευάζουμε μία κλάση που κληρονομεί την json.JSONDecoder
  - Στην οποία ορίζουμε στον κατασκευαστή της, τη μέθοδο αποκωδικοποίησης, ως όρισμα με λέξη κλειδί
- Καλείται η load[s] με όρισμα στην παράμετρο με λέξη-κλειδί **cls** το όνομα της παραπάνω κλάσης.

Παράδειγμα 7: json_objects_manip_class

```
class Time:
    ...
```

```
class TimeDecoder(json.JSONDecoder):
    def __init__(self):
        super().__init__(object_hook=self.decoder)

    def decoder(self, time):
        return Time(time["hour"],
                    time["minute"],
                    time["second"])
```

```
with open("ob.json", "r") as f:
    t2 = json.load(f, cls=TimeDecoder)
    print(t2)
```

- Για να αποθηκεύσουμε **πολλά αντικείμενα του ίδιου τύπου**, μπορούμε:
  - είτε να τα αποθηκεύουμε ανά γραμμή, όπως στο Μαθ.15 και έπειτα να τα διαβάσουμε γραμμή - γραμμή
  - είτε να τα ενσωματώσουμε σε μία δομή - περιτυλιχτή (π.χ. μια λίστα) και να τα διαβάσουμε όλα μαζί, ορίζοντας ότι τα JSON αντικείμενα θα μετατρέπονται σε αντικείμενα της κλάσης μας με κατάλληλο μετατροπέα.

Παράδειγμα 8: json_many_objects_same_type

```
from timeclass import Time, time_encoder, time_decoder
from random import randrange
import json
arr = []
for i in range(10):
    t = Time(randrange(24), randrange(60), randrange(60))
    arr.append(t)
with open("ob.json", "w") as f:
    json.dump(arr, f, default=time_encoder)

with open("ob.json", "r") as f:
    data = json.load(f, object_hook=time_decoder)
    for t in data:
        print(t)
```

- Για να αποθηκεύσουμε **αντικείμενα διαφορετικών τύπων**
  - αποθηκεύουμε στο JSON αντικείμενο έξτρα πληροφορία για την κλάση που ορίζεται από το αντικείμενο στην μετατροπή του σε JSON
  - προσθέτουμε λειτουργικότητα στον αποκωδικοποιητή ώστε να αναγνωρίζει τον τύπο του αντικειμένου και να καλεί τον κατάλληλο κατασκευαστή.

Παράδειγμα 9: json_many_objects_different_types

```
def time_encoder(time):
    if isinstance(time, ZonedTime):
        return {"__ZonedTime__": True, "hour": time.hour,
                "minute": time.minute, "second": time.second,
                "zone": time.zone}
    elif isinstance(time, Time):
        return {"__Time__": True, "hour": time.hour, "minute": time.minute,
                "second": time.second}
    else:
        raise TypeError("Type should be Time")

def time_decoder(time):
    if "__Time__" in time:
        return Time(time["hour"], time["minute"], time["second"])
    elif "__ZonedTime__" in time:
        return ZonedTime(time["hour"], time["minute"], time["second"],
time["zone"])
```