Programme du jour

- Constructeurs de Classes
- Listes
- JSON loading/parsing/saving
- built-int functions
- date
- serializer

Sujet de travail

Pour rester dans la thématique des animaux, aujourd’hui nous construirons des fermes. Nous les administrerons et peut être même on pourra les envoyer par email !

Description de la ferme

```
class Animal

    type = “Vache”
    birth_year =
    sex =
    ....


class Farm :

    def __init__(self, name):

    def __init__(self, name, json_string):

    def __str__(self):
```

```
Farm_list = []

Farm_list.append(Farm(“Première ferme”))

Farm_list.append(Farm(“Deuxième ferme”, json string)

Farm_list[0].add(Animal(...))

print(Farm[0])
```

Manipulation de listes

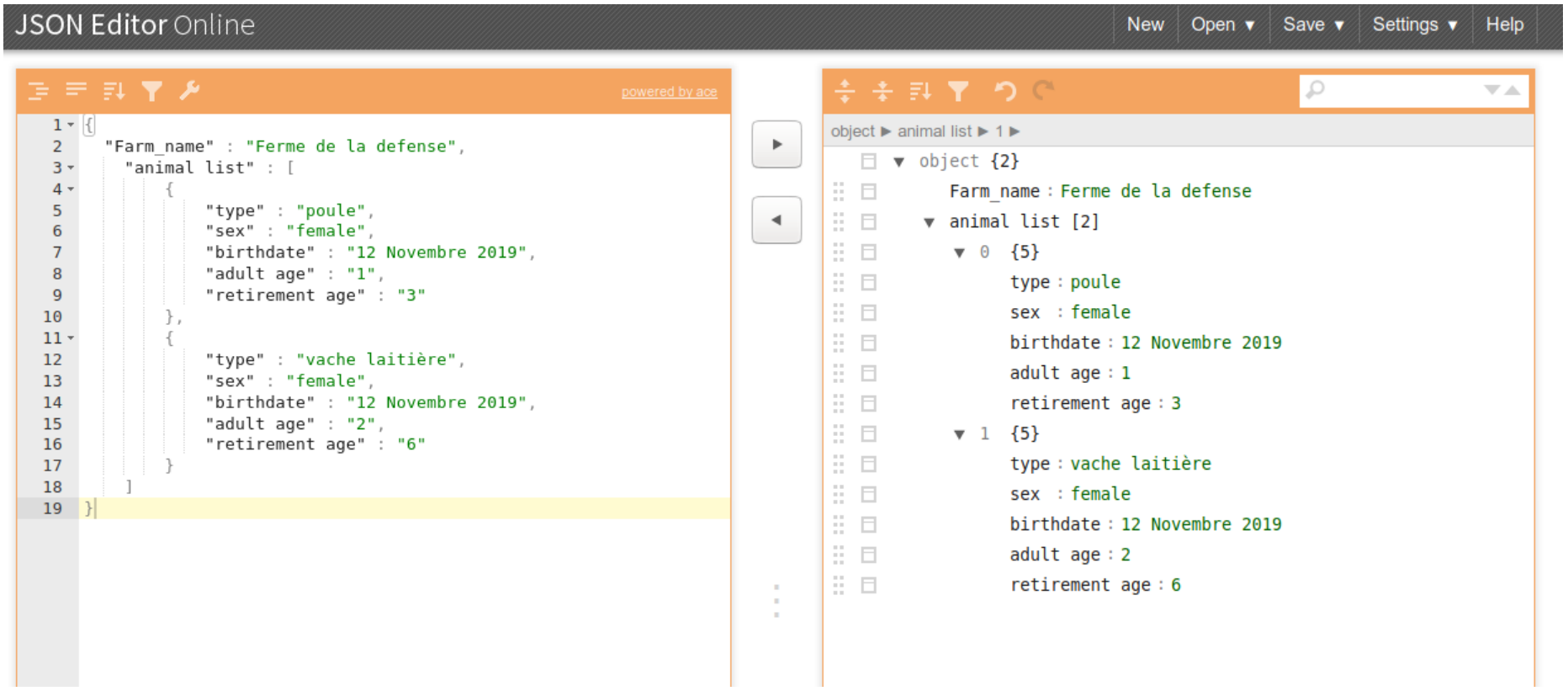
- append()
- insert()
- index()
- count()
- sort()
- copy : “=”
- pop()

Json

```
import json

with open('data.txt') as json_file:
    data = json.load(json_file)

with open('data.txt', 'w') as outfile:
    json.dump(data, outfile)

json.dumps(data, indent=4)
```

=> utiliser l’interpréteur python pour comprendre comment naviguer dans l’objet Json

=> pprint()


JSON Editor Online
New
Open
Save
Settings
Help
powered by ace
{
"Farm_name" : "Ferme de la defense",
"animal list" : [
{
"type" : "poule",
"sex" : "female",
"birthdate" : "12 Novembre 2019",
"adult age" : "1",
"retirement age" : "3"
},
{
"type" : "vache laitière",
"sex" : "female",
"birthdate" : "12 Novembre 2019",
"adult age" : "2",
"retirement age" : "6"
}
]
}
object ▶ animal list ▶ 1 ▶
object {2}
Farm_name : Ferme de la defense
animal list [2]
0 {5}
type : poule
sex : female
birthdate : 12 Novembre 2019
adult age : 1
retirement age : 3
1 {5}
type : vache laitière
sex : female
birthdate : 12 Novembre 2019
adult age : 2
retirement age : 6

Et le temps passe ...

- Les animaux vieillissent
- Les animaux qui sont adultes se reproduisent (1 Male + 1 femelle = 1 nouvel animal, utiliser rand())
- Les animaux qui doivent “partir à la retraite” sont retiré de la ferme et envoyé dans une ferme “sanctuaire” ou le temps ne s’écoule plus

builtin functions

- __add__()
- __iadd__()
- __len__()
- __eq__()
- __delattr__()

https://openclassrooms.com/fr/courses/235344-apprenez-a-programmer-en-python/233046-appliquez-des-methodes-speciales

Première partie

1) Ecrire les classes avec toutes les builtin functions
2) Faire un main qui crée une liste de 3 fermes et y rajoute quelques animaux
3) Charger une ferme à partir d’un fichier Json
4) Imprimer la liste de la ferme
5) additionner deux fermes (fusionner), soustraire deux fermes ?
6) Faire passer le temps pour voir ce qu’il se passe (rajouter mois par mois, utiliser l’interpréteur)

Serialisation

```
import cPickle
text = cPickle.dumps(data)
bytes = cPickle.dumps(data, 1)
redata = cPickle.loads(bytes)
ouf = open('datafile.txt', 'w')
cPickle.dump(data, ouf)
ouf.close(  )
```

Documentation de votre code

Comme vous allez partager votre code avec vos voisins, il est nécessaire de bien le commenter en utilisant les docstring. Il pourra essayer de comprendre vos objets **sans lire le code** en utilisant help().

Deuxième partie : Sérialisation

- Serialiser votre liste de fermes
- l’envoyer par email à votre voisin (ou clé usb)
- votre voisin la désérialise, et fait des transferts d’animaux
- Que se passe t’il ? Faire en sorte de rendre possible le transfert d’animaux SANS que les objets fermes soient identiques !!!
- Additionner deux fermes ? ;)