PAGE:
DATE: / /

Mission Name:- List Comprehensions And Lambda Functions

JSON :- JavaScript Object Notation

Python JSON module :- import json
Loading JSON dataset :- json.loads()

Imp → json.loads() function, loads the json data from a string.
→ json.load() function, loads from a file object.

* Opening and loading a json file:-

```
file = open("hn_2014.json")
hn = json.load(file)
```

→ json.dumps() → returns the string version of object.
Syntax:-

```
text = json.dumps(obj, sort_keys=True, indent=4)
```

string version — json object — dictionary key sort

* del keyword can be used to delete a dictionary key.
Eg:-

```
def del_key(dict_, key):
    modified_dict = dict_.copy()
    del modified_dict[key]
    return modified_dict

hn_clean = []
for _ in hn:
    hn_clean.append(del_key(_, "createdAt"))
```

PAGE:
DATE: / /

* List Comprehension:-

Loop version:-

```
rounded = []
for f in floats:
    rounded.append(round(f))
```

Target variable

for statement

transformation.

List Comprehension version:-

```
rounded = [round(f) for f in floats]
```

target variable

transformation

for statement

A list comprehension can be used where we:-
i) Iterate over values in list.
ii) Performed a transformation on those values.
iii) Assigned the result to a new list.

Eg:- urls = [_["url"] for _ in hn_clean]

Eg2:- thousand_points = [p for p in hn_clean if p["points"]>1000]

* [If we omit parenthesis from the function name, we can use it as variable.]

* Passing functions as arguments:-

Eg:-
```
def key_func(d):
    return d["numComments"]
most_comments = max(hn_clean, key=key_func)
```

we passed key_func as argument.

PAGE:
DATE: / /

* Lambda Function :-

Python provides a special syntax to create temporary functions for situations when we need to use the funct only once. These functions are called lambda functions.

Eg 1:-
```
def unchanged(x):
    return x
```
function name → unchanged, Parameter → x, transformation → return x

Lambda Equivalent :-
```
unchanged = lambda x : x
```

Eg 2:-
```
def add(x,y):
    return x+y
```

Lambda Equivalent :-
```
add = lambda x,y : x+y
```

* Pandas function to read json files/strings :-

```
pandas.read_json()
```

*
```
tags = hn_df["tags"]
mask = tags.apply(len)==4      # boolean mask
four_tags = tags[mask]
```

*
```
if len(l)==4:
    return l[-1]
else:
    return None
```

Ternary Equivalent :-

```
l[-1] if len(l)==4 else None
```