Modul 9 Praktikum Pemrograman Berbasis Fungsi

April 16, 2023

Tujuan Praktikum

1. Mahasiswa dapat menerapkan data preprocessing dengan functional programming
2. Mahasiswa dapat menggunakan itertools modul

Jangan lupa untuk menginstall: `!pip install maze`

0.1 Naive

0.1.1 Reading data

[ ]:
```
from urllib.request import urlopen
from json import loads

BASE = 'https://api.github.com/search'
_url1 = '{}/repositories?q={}'
q = 'data&per_page=100'
url1 = _url1.format(BASE, q)
f = urlopen(url1)
data = loads(f.read().decode('utf-8'))
repos = data['items']
repos[0]['description']
```

[ ]:
```
'Data and code behind the articles and graphics at FiveThirtyEight'
```

[ ]:
```
repos[0]['full_name']
```

[ ]:
```
'fivethirtyeight/data'
```

0.1.2 Processing data

[ ]:
```
def rate(repos):
    rated = []

    for repo in repos:
        rated.append(repo['watchers'] * 2)

    return rated
```

[ ]:
```
rate(repos)[:5]
```

[ ]:
```
[32312, 6054, 8856, 3904, 2968]
```

[ ]:
```
# Infinite data
from itertools import count

inf_repos = ({'watchers': c} for c in count())

# Don't actually run the below code since it will hang forever
# rate(inf_repos)
```

[ ]:
```
# Expensive data
from time import sleep

def exp_rate(repos):
    rated = []

    for repo in repos:
        sleep(1)
        rated.append(repo['watchers'] * 2)

    return rated
```

[ ]:
```
exp_rate(repos)[:5]
```

[ ]:
```
[32312, 6054, 8856, 3904, 2968]
```

0.2 Lazy evaluation

[ ]:
```
eager_list = list(range(5))
eager_list
```

[ ]:
```
[0, 1, 2, 3, 4]
```

[ ]:
```
lazy_list = iter(eager_list)
lazy_list
```

[ ]:
```
<list_iterator at 0x1c45163c310>
```

[ ]:
```
next(lazy_list)
```

[ ]:
```
0
```

[ ]:
```
list(lazy_list)
```

[ ]:
```
[1, 2, 3, 4]
```

Di bawah ini sudah terjadi stopiteration pada next lazylist

```
[ ]: next(lazy_list)
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
Cell In[12], line 1
----> 1 next(lazy_list)

StopIteration:
```

0.2.1 Reading data

```
[ ]: from ijson import items

     f = urlopen(url1)
     repos = items(f, 'items.item')
     repos
```

```
[ ]: <generator object items at 0x000001C4527CD9C0>
```

```
[ ]: repo = next(repos)
     repo['full_name']
```

```
[ ]: 'fivethirtyeight/data'
```

0.2.2 Processing data

```
[ ]: def gen_rates(repos):
         for repo in repos:
             yield repo['watchers'] * 2
```

```
[ ]: gen_rates(repos)
```

```
[ ]: <generator object gen_rates at 0x000001C45161A670>
```

```
[ ]: rates = gen_rates(repos)
     next(rates)
```

```
[ ]: 6054
```

```
[ ]: next(rates)
```

```
[ ]: 8856
```

```
[ ]: # Infinite data
     rates = gen_rates(inf_repos)
     next(rates)
```

[ ]: 0

[ ]:
```
# Expensive data
def gen_exp_rates(repos):
    for repo in repos:
        sleep(1)
        yield repo['watchers'] * 2
```

[ ]:
```
from itertools import islice

rates = gen_exp_rates(repos)
result = islice(rates, 5)
list(result)
```

[ ]:
```
[3904, 2968, 1134, 562, 1758]
```

[ ]:
```
next(rates)
```

[ ]: 844

0.3 Grouping data

[ ]:
```
f = urlopen(url1)
repos = items(f, 'items.item')
repo = next(repos)
repo.keys()
```

[ ]:
```
dict_keys(['id', 'node_id', 'name', 'full_name', 'private', 'owner', 'html_url',
'description', 'fork', 'url', 'forks_url', 'keys_url', 'collaborators_url',
'teams_url', 'hooks_url', 'issue_events_url', 'events_url', 'assignees_url',
'branches_url', 'tags_url', 'blobs_url', 'git_tags_url', 'git_refs_url',
'trees_url', 'statuses_url', 'languages_url', 'stargazers_url',
'contributors_url', 'subscribers_url', 'subscription_url', 'commits_url',
'git_commits_url', 'comments_url', 'issue_comment_url', 'contents_url',
'compare_url', 'merges_url', 'archive_url', 'downloads_url', 'issues_url',
'pulls_url', 'milestones_url', 'notifications_url', 'labels_url',
'releases_url', 'deployments_url', 'created_at', 'updated_at', 'pushed_at',
'git_url', 'ssh_url', 'clone_url', 'svn_url', 'homepage', 'size',
'stargazers_count', 'watchers_count', 'language', 'has_issues', 'has_projects',
'has_downloads', 'has_wiki', 'has_pages', 'has_discussions', 'forks_count',
'mirror_url', 'archived', 'disabled', 'open_issues_count', 'license',
'allow_forking', 'is_template', 'web_commit_signoff_required', 'topics',
'visibility', 'forks', 'open_issues', 'watchers', 'default_branch', 'score'])
```

[ ]:
```
repo['has_issues']
```

[ ]: True

```
[ ]: import itertools as it
     from operator import itemgetter

     keyfunc = itemgetter('has_issues')
     sorted_repos = sorted(repos, key=keyfunc)
     grouped = it.groupby(sorted_repos, keyfunc)
     data = ((key, len(list(group))) for key, group in grouped)
     next(data)
```

```
[ ]: (False, 6)
```

```
[ ]: next(data)
```

```
[ ]: (True, 93)
```

0.4 Memoization

0.4.1 Processing data

```
[ ]: def calc_rate(watchers):
         sleep(1)
         return watchers * 2

     def gen_exp_rates(repos):
         for repo in repos:
             yield calc_rate(repo['watchers'])
```

```
[ ]: repos = it.repeat({'watchers': 5})
     rates = gen_exp_rates(repos)
     result = islice(rates, 5)
     list(result)
```

```
[ ]: [10, 10, 10, 10, 10]
```

```
[ ]: from functools import lru_cache

     def _calc_rate(watchers):
         sleep(1)
         return watchers * 2

     cacher = lru_cache()
     calc_rate = cacher(_calc_rate)

     def gen_exp_rates(repos):
         for repo in repos:
             yield calc_rate(repo['watchers'])
```

```
[ ]: repos = it.repeat({'watchers': 5})
     rates = gen_exp_rates(repos)
     result = islice(rates, 5)
     list(result)
```

```
[ ]: [10, 10, 10, 10, 10]
```

```
[ ]: @lru_cache()
     def calc_rate(watchers):
         sleep(1)
         return watchers * 2

     def gen_exp_rates(repos):
         for repo in repos:
             yield calc_rate(repo['watchers'])
```

```
[ ]: repos = it.repeat({'watchers': 5})
     rates = gen_exp_rates(repos)
     result = islice(rates, 5)
     list(result)
```

```
[ ]: [10, 10, 10, 10, 10]
```

0.5 Introducing meza

0.5.1 Reading data

```
[ ]: from urllib.request import urlopen
     from meza.io import read_json

     url2 = '{}/repositories?q=data'.format(BASE)
     f = urlopen(url2)
     records = read_json(f, path='items.item')
     repo = next(records)
     repo['full_name']
```

```
[ ]: 'fivethirtyeight/data'
```

```
[ ]: len(list(records))
```

```
[ ]: 29
```

```
[ ]: from io import StringIO
     from meza.io import read_csv

     f = StringIO('greeting,location\nhello,world\n')
     next(read_csv(f))
```

```
[ ]: {'greeting': 'hello', 'location': 'world'}
```

```
[ ]: from os import path as p
     from meza.io import join

     url3 = '{}&page=2'.format(url2)
     files = map(urlopen, [url2, url3])
     records = join(*files, ext='json', path='items.item')
     repo = next(records)
     repo['full_name']
```

```
[ ]: 'fivethirtyeight/data'
```

```
[ ]: repo['language']
```

```
[ ]: 'Jupyter Notebook'
```

```
[ ]: len(list(records))
```

```
[ ]: 59
```

0.5.2 Transforming data

```
[ ]: from meza.process import merge

     records = [{'a': 200}, {'b': 300}, {'c': 400}]
     merge(records)
```

```
[ ]: {'a': 200, 'b': 300, 'c': 400}
```

```
[ ]: from meza.process import group

     records = [
         {'item': 'a', 'amount': 200},
         {'item': 'a', 'amount': 200},
         {'item': 'b', 'amount': 400}]

     grouped = group(records, 'item')
     key, _group = next(grouped)
     key
```

```
[ ]: 'a'
```

```
[ ]: _group
```

```
[ ]: [{'item': 'a', 'amount': 200}, {'item': 'a', 'amount': 200}]
```

[ ]:
```
from meza import process as pr

f = urlopen(url2)
raw = read_json(f, path='items.item')
fields = ['full_name', 'language', 'watchers', 'score', 'has_wiki']
cut = pr.cut(raw, fields)
cut
```

[ ]:
```
<generator object cut.<locals>.<genexpr> at 0x000001C45280D140>
```

[ ]:
```
cut, preview = pr.peek(cut)
cut
```

[ ]:
```
<itertools.chain at 0x1c451d932b0>
```

[ ]:
```
len(preview)
```

[ ]:
```
5
```

[ ]:
```
preview[0]
```

[ ]:
```
{'full_name': 'fivethirtyeight/data',
 'language': 'Jupyter Notebook',
 'has_wiki': True,
 'watchers': 16156,
 'score': Decimal('1.0')}
```

[ ]:
```
filled = pr.fillempty(raw, value='', fields=['language'])
pivoted = pr.pivot(filled, 'score', 'language', rows=['has_wiki'], op=min)
next(pivoted)
```

[ ]:
```
{'JavaScript': Decimal('1.0'),
 'has_wiki': False,
 'Python': Decimal('1.0'),
 'TypeScript': Decimal('1.0')}
```

[ ]:
```
next(pivoted)
```

[ ]:
```
{'': Decimal('1.0'),
 'has_wiki': True,
 'C++': Decimal('1.0'),
 'CSS': Decimal('1.0'),
 'HTML': Decimal('1.0'),
 'Java': Decimal('1.0'),
 'JavaScript': Decimal('1.0'),
 'Jupyter Notebook': Decimal('1.0'),
 'PHP': Decimal('1.0'),
 'Python': Decimal('1.0'),
 'R': Decimal('1.0'),
 'Vue': Decimal('1.0')}
```