Московский государственный технический университет им. Н.Э. Баумана

Факультет ИУ Кафедра ИУ5

Курс «Основы информатики»
Отчет по лабораторной работе №2

Выполнил студент группы ИУ5-33Б: Емельянов А.К. Подпись и дата:

Проверил преподаватель каф.: Гапанюк Ю. Е. Подпись и дата:

Москва, 2024 г

Цель лабораторной работы: изучение возможностей функционального программирования в языке Python.

№1

```python
def field(items, *args):
    assert len(args) > 0

    if len(args) == 1:
        for item in items:
            if args[0] in item and item[args[0]] is not None:
                yield item[args[0]]
    else:
        for item in items:
            dict = {}
            all_none = True
            for key in args:
                if key in item and item[key] is not None:
                    dict[key] = item[key]
                    all_none = False
            if not all_none:
                yield dict

goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
]

print(str(list(field(goods, 'title')))[1:-1])
print(str(list(field(goods, 'title', 'price')))[1:-1])
```

```
acti0n@MacBook-Alexey proga % /usr/local/bin/python3 /Users/acti0n/Documents/pr
a_1.py
'Ковер', 'Диван для отдыха'
{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}
acti0n@MacBook-Alexey proga %
```

№2

```python
from random import randint
def gen_random(num_count, begin, end):
    ans = []
    for i in range(num_count):
        ans.append(randint(begin,end))
    yield ans


print(str(list(gen_random(5,1,3)))[1:-1])
```

```
acti0n@MacBook-Alexey proga % /usr/local/bin/python3 /Users/acti0n/Documents/pr
a_2.py
[3, 1, 3, 2, 1]
acti0n@MacBook-Alexey proga %
```

№3

```python
from test import gen_random


class Unique(object):
    def __init__(self, items, **kwargs):
        self.data = iter(items)
        self.ignore_case = kwargs.get('ignore_case', False)
        self.unique_items = set()

    def __next__(self):
        while True:
            item = next(self.data)
            check_item = item.lower() if self.ignore_case else item
            if check_item not in self.unique_items:
                self.unique_items.add(check_item)
                return item

    def __iter__(self):
        return self
```

```python
data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
unique_data = Unique(data)
print(list(unique_data))


data = gen_random(10, 1, 3)
unique_data = Unique(data)
print(list(unique_data))


data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
unique_data = Unique(data)
print(list(unique_data))


unique_data_ignore_case = Unique(data, ignore_case=True)
print(list(unique_data_ignore_case))
```

```
a_3.py
[1, 2]
[1, 3, 2]
['a', 'A', 'b', 'B']
['a', 'b']
acti0n@MacBook-Alexey proga %
```

№4

```python
data = [4, -30, 100, -100, 123, 1, 0, -1, -4]


if __name__ == '__main__':
    result =  sorted(data, key=lambda x: abs(x), reverse=True)
    print(result)


    result_with_lambda = sorted(data, key=abs, reverse=True)
    print(result_with_lambda)
```

```
a_4.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
acti0n@MacBook-Alexey proga %
```

№5

```python
def print_result(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)

        if isinstance(result, list):
            for item in result:
                print(item)

        elif isinstance(result, dict):
            for key, value in result.items():
                print(f'{key} = {value}')

        else:
            print(result)

        return result

    return wrapper

@print_result
def test_1():
    return 1


@print_result
def test_2():
    return 'iu5'


@print_result
def test_3():
    return {'a': 1, 'b': 2}


@print_result
def test_4():
    return [1, 2]
```

```
if __name__ == '__main__':
    test_1()
    test_2()
    test_3()
    test_4()
```

```
a_5.py
1
iu5
a = 1
b = 2
1
2
acti0n@MacBook-Alexey proga %
```

№6

```
import time
from contextlib import contextmanager

class cm_timer_1:
    def __enter__(self):
        self.start_time = time.time()

    def __exit__(self, exc_type, exc_val, exc_tb):
        elapsed_time = time.time() - self.start_time
        print(f"time: {elapsed_time}")

@contextmanager
def cm_timer_2():
    start_time = time.time()
    yield
    elapsed_time = time.time() - start_time
    print(f"time: {elapsed_time}")

# Использование cm_timer_1
with cm_timer_1():
    time.sleep(5.5)
```

```python
# Использование cm_timer_2
with cm_timer_2():
    time.sleep(5.5)
```

```
a_6.py
time: 5.5048980712890625
time: 5.504918098449707
acti0n@MacBook-Alexey proga %
```

№7

```python
import json
from test import gen_random
from Zadacha_5 import print_result
from Zadacha_6 import cm_timer_1


path = "/Users/acti0n/Documents/proga/lab2/data.json"

with open(path, encoding='utf-8') as f:
    data = json.load(f)

@print_result
def f1(arg):
    return sorted(set(item['job-name'].lower() for item in arg))

@print_result
def f2(arg):
    return list(filter(lambda s: s.startswith('программист'), arg))

@print_result
def f3(arg):
    return list(map(lambda s: s + ' с опытом Python', arg))

@print_result
def f4(arg):
    salaries = gen_random(len(arg), 100000, 2000000)
    return ['{} зарплата {}'.format(job, salary) for job, salary in zip(arg, salaries)]
```

```
if __name__ == '__main__':
    with cm_timer_1():
        f4(f3(f2(f1(data))))
```

ТЕРМИНАЛ

```
электросварщик на автоматических и полуавтоматических машинах
электросварщик ручной сварки
электросварщики ручной сварки
электрослесарь (слесарь) дежурный и по ремонту оборудования, старший
электрослесарь по ремонту и обслуживанию автоматики и средств измерений электрост
электрослесарь по ремонту оборудования в карьере
электроэрозионист
эндокринолог
энергетик
энергетик литейного производства
энтомолог
юрисконсульт
юрисконсульт 2 категории
юрисконсульт. контрактный управляющий
юрист
юрист (специалист по сопровождению международных договоров, английский – разговор
юрист волонтер
юристконсульт
программист
программист / senior developer
программист 1с
программист с#
программист с++
программист с++/с#/java
программист/ junior developer
программист/ технический специалист
программистр-разработчик  информационных систем
программист с опытом Python
программист / senior developer с опытом Python
программист 1с с опытом Python
программист с# с опытом Python
программист с++ с опытом Python
программист с++/с#/java с опытом Python
программист/ junior developer с опытом Python
программист/ технический специалист с опытом Python
программистр-разработчик  информационных систем с опытом Python
программист с опытом Python зарплата 1909229
программист / senior developer с опытом Python зарплата 1973342
программист 1с с опытом Python зарплата 1793334
программист с# с опытом Python зарплата 1724839
программист с++ с опытом Python зарплата 609915
программист с++/с#/java с опытом Python зарплата 1808399
программист/ junior developer с опытом Python зарплата 753217
программист/ технический специалист с опытом Python зарплата 1467939
программистр-разработчик  информационных систем с опытом Python зарплата 1992503
```