Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра «Информатика и вычислительная техника»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №3

Выполнил:
студент группы РТ5-31:
Слкуни Г.Г.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2023

Текст программы:

cm_timer.py:

```
from contextlib import contextmanager
from time import time, sleep

# С помощью библиотеки
@contextmanager
def cm_timer_2():
    start = time()
    yield lambda: time() - start
    print('time: {}'.format(time() - start))

# С помощью класса
class cm_timer_1():
    def __enter__(self):
        self.start = time()
        return self

    def __exit__(self, type, value, traceback):
        print('time: {}'.format(time() - self.start))

def main():
    with cm_timer_1():
        sleep(1)

    with cm_timer_2():
        sleep(1)

if __name__ == '__main__':
    main()
```

field.py:

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

def field(items, *args):
    result = []
    for item in items:
        constructed_item = {}
        for key in args:
            if key in item:
                constructed_item[key] = item[key]
        if constructed_item:
            result.append(constructed_item)
    return result
```

```
def main():
    print(field(goods))
    print(field(goods, 'title'))
    print(field(goods, 'title', 'price'))

if __name__ == '__main__':
    main()
```

gen_random.py:

```
import random

def gen_random(amount, min, max):
    return [random.randint(min, max) for i in range(amount)]

def main():
    print(gen_random(5, 1, 3))

if __name__ == '__main__':
    main()
```

print_result.py:

```
def print_result(func):
    def wrapper(*args, **kwargs):
        print(func.__name__)
        result = func(*args, **kwargs)
        if isinstance(result, list):
            [print(element) for element in result]
        elif isinstance(result, dict):
            [print('{} = {}'.format(key, value)) for key, value in
result.items()]
        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()
```

sort.py:

```
def main():
    data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
    # Без lambda-функции
    print(sorted(data, key=abs, reverse=True))
    # С lambda-функцией
    print(sorted(data, key=lambda x: abs(x), reverse=True))

if __name__ == '__main__':
    main()
```

uniqiue.py:

```
from gen_random import gen_random

class Unique(object):
    def __init__(self, items, **kwargs):
        self.ignore_case = 'ignore_case' in kwargs and kwargs['ignore_case']
        self.items = items
        self.originals = []
        self.item_index = -1
        pass

    def __next__(self):
        self.item_index += 1
        if self.item_index >= len(self.items): raise StopIteration
        is_item_string = isinstance(self.items[self.item_index], str)
        if (not self.items[self.item_index] in self.originals and
(self.ignore_case and is_item_string or not is_item_string)) or (is_item_string
and not self.ignore_case and len(list(filter(lambda item: isinstance(item, str)
and item.lower() == self.items[self.item_index].lower(), self.originals))) == 0):
            self.originals.append(self.items[self.item_index])
            return self.items[self.item_index]
        return self.__next__()

    def __iter__(self):
```

```
        return self

def main():
    unique = Unique(['A', 'B', 'a', 'b', 'c', 'c'], ignore_case=False)
    for element in unique:
        print(element, end=' ')
    print('')
    unique = Unique(gen_random(10, 1, 6))
    for element in unique:
        print(element, end=' ')

if __name__ == '__main__':
    main()
```

process_data.py:

```
import json

# Импортируем предыдущие микрозадачи
from print_result import print_result
from cm_timer import cm_timer_1
from unique import Unique
from gen_random import gen_random

path = './data_light.json'

@print_result
def f1(arg):
   return sorted([x['job-name'] for x in Unique(arg)])

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

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

@print_result
def f4(arg):
    salaries = gen_random(len(arg), 100000, 200000)
    return ['{}, зарплата {} руб.'.format(job, salary) for job, salary in
zip(arg, salaries)]

def main():
    with open(path, encoding='utf-8') as f:
        data = json.load(f)
        f.close()
    with cm_timer_1():
        f4(f3(f2(f1(data))))
```

```
if __name__ == '__main__':
    main()
```

Результат выполнения:

cm_timer.py:

time: 1.0040361881256104

time: 1.0128965377807617

field.py:

[]

[{'title': 'Ковер'}, {'title': 'Диван для отдыха'}]

[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха'}]

gen_random.py:

[3, 3, 1, 3, 1]

print_result.py:

test_1

1

test_2

iu5

test_3

a = 1

b = 2

test_4

1

2

sort.py:

[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]

[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]

unique.py:

A B c

4 1 3 6 2

process_data.py:

f1

1C программист

2-ой механик

3-ий механик

4-ый механик

4-ый электромеханик

ASIC специалист

…

программист с опытом Python, зарплата 182386 руб.

программист с опытом Python, зарплата 195578 руб.

программист с опытом Python, зарплата 103220 руб.

программист 1C с опытом Python, зарплата 123453 руб.

time: 1.4823293685913086