**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: |  | Проверил: |
| студент группы РТ5-31: |  | преподаватель каф. ИУ5 |
| Слкуни Г.Г. |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | 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

1С 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 1С с опытом Python, зарплата 123453 руб.

time: 1.4823293685913086