

**Московский государственный технический
университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Парадигмы и конструкции языков
программирования»

Отчет по лабораторной работе №2

Выполнил: Сорокин Михаил
Группа ИУ5-33Б
Вариант 15 В

Проверил: Гапанюк Юрий
Евгеньевич
преподаватель каф. ИУ5

Подпись и дата: 23.10.2024




Подпись и дата:

Москва, 2024 г.


Принт-скрин выполнения программы

Cm_timer

 D:\anaconda\python.exe


```
5.5135
5.5026
None
Press any key to continue . . .
```

Field

 D:\anaconda\python.exe


```
Carpet Sofa
Carpet 2000 Sofa 5300
Press any key to continue . . .
```

Gen_random

 D:\anaconda\python.exe


```
1
1
1
1
1
1
2
Press any key to continue
```

Print_result

 D:\anaconda\python.exe


```
!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
Press any key to continue . . .
```

Sort

 D:\anaconda\python.exe

```
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
Press any key to continue . . .
```

Unique

 D:\anaconda\python.exe

```
<__main__.Unique object at 0x000001D6E052D910>
<__main__.Unique object at 0x000001D6E052D970>
Press any key to continue . . .
```

Листинг

Пакет lab_python_fp модуль cm_timer.py

```
import contextlib
import time
import sys

class cm_timer_1(object):
    def __enter__(self):
        self.t = time.perf_counter()
        return self

    def __exit__(self, type, value, traceback):
        self.e = time.perf_counter()

    def __float__(self):
        return float(self.e - self.t)

    def __str__(self):
        return f"{self.e - self.t:.4f}"

@contextlib.contextmanager
def cm_timer_2():
    start_time = time.perf_counter()
    yield
    end_time = time.perf_counter()
    print(f"{end_time - start_time:.4f}")

def main():
    with cm_timer_1() as t1:
        time.sleep(5.5)
    print(t1)

    with cm_timer_2() as t2:
        time.sleep(5.5)
    print(t2)

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

модуль field.py

```
def field(items, *args):
    assert len(args) > 0
    for item in items:
        for arg in args:
            if item[arg]:
                print(item[arg], end = ' ')
        print()

def main():
    goods = [
        {'title': 'Carpet', 'price': 2000, 'color': 'green'},
        {'title': 'Sofa', 'price': 5300, 'color': 'black'}
    ]
    field(goods, 'title')
    field(goods, 'title', 'price')

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

модуль gen_random.py

```
import random
```

```

def gen_random(num_count, begin, end):
    for _ in range(num_count):
        print(random.randrange(begin, end))

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

if __name__ == "__main__":
    main()

```

модуль print_result.py

```

def print_result(input_func):
    def output_func():
        print(input_func.__name__)

        result = input_func()
        if isinstance(result, list):
            for elem in result:
                print(elem)
        elif isinstance(result, dict):
            for key in result:
                print(f"{key} = {result[key]}")
        else:
            print(result)
    return output_func

@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]

def main():
    print('!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()

if __name__ == "__main__":
    main()

```

модуль process_data.py

```

import json
import sys

path = None

with open(path) as f:
    data = json.load(f)

```

```
@print_result
def f1(arg):
    raise NotImplemented
```

```
@print_result
def f2(arg):
    raise NotImplemented
```

```
@print_result
def f3(arg):
    raise NotImplemented
```

```
@print_result
def f4(arg):
    raise NotImplemented
```

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

модуль sort.py

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

    def sort_by_abs(x):
        return abs(x)
    result_with_lambda = sorted(data, key = sort_by_abs, reverse = True)
    print(result_with_lambda)

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

модуль unique.py

```
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items = False, **kwargs):
        # Нужно реализовать конструктор
        self.ignore_case = items
        self.dict = kwargs

    def __next__(self):
        # Нужно реализовать __next__
        pass

    def __iter__(self):
        return self

def main():
    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    print(Unique(data))
    print(Unique(data, ignore_case=True))

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