Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №3

«**Функциональные возможности языка Python.**»

Выполнил
Студент группы ИУ5-31Б
Ван
Чжувэнь

Проверил:
Гапанюк Ю.Е.

2025 г.

Листинг кода

cm_timer.py

```
import time
from contextlib import contextmanager

# Способ 1: На основе класса
class cm_timer_1:
    def __enter__(self):
        self.start_time = time.time()
        return self

    def __exit__(self, exc_type, exc_val, exc_tb):
        elapsed_time = time.time() - self.start_time
        print(f'time: {elapsed_time:.2f}')

# Способ 2: С использованием библиотеки contextlib
@contextmanager
def cm_timer_2():
    start_time = time.time()
    try:
        yield
    finally:
        elapsed_time = time.time() - start_time
        print(f'time: {elapsed_time:.2f}')

print("\nЗадание 6")
print("Тест cm_timer_1:")
with cm_timer_1():
    time.sleep(2.5)

print("\nТест cm_timer_2:")
with cm_timer_2():
    time.sleep(1.5)
```

field.py

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

    if len(args) == 1:
        # Если передан один аргумент - возвращаем только значения этого поля
        key = args[0]
        for item in items:
            value = item.get(key)
            if value is not None:
                yield value
    else:
        # Если передано несколько аргументов - возвращаем словари
        for item in items:
            result = {}
            for key in args:
                value = item.get(key)
```

```
                if value is not None:
                    result[key] = value
            # Если хотя бы одно поле не None, выдаем словарь
            if result:
                yield result

print("Задание 1")
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'color': 'black'}
]

print("Тест 1 - один аргумент 'title':")
for value in field(goods, 'title'):
    print(value)

print("\nТест 2 - один аргумент 'price':")
for value in field(goods, 'price'):
    print(value)

print("\nТест 3 - два аргумента 'title', 'price':")
for value in field(goods, 'title', 'price'):
    print(value)
```

gen_random.py

```
# run_tests.py
import unittest
import sys

if __name__ == '__main__':
    # Загрузка тестов из модуля test_rk1
    test_loader = unittest.TestLoader()
    test_suite = test_loader.discover('.', pattern='test_*.py')

    # Запуск тестов
    test_runner = unittest.TextTestRunner(verbosity=2)
    result = test_runner.run(test_suite)

    # Возврат кода выхода в зависимости от результата тестов
    sys.exit(0 if result.wasSuccessful() else 1)
```

main.py

```
import json
import sys
from field import field
from gen_random import gen_random
from unique import Unique
from sort import sort
from print_result import print_result
from cm_timer import cm_timer_1
```

```
import random

from typing import List, Dict, Any

with open("data.json", 'r', encoding='utf-8') as file:
    json_data = json.load(file)

def unique(data):
    unique_data = set()
    for dct in data:
        job = dct.get("job-name")
        if job and job not in unique_data:
            yield job
            unique_data.add(job)

def get_jobs(data: List[Dict]) -> List:
    return list(unique(data))

def unique_reverse_sort(data: List):
    res = sorted(data, key = lambda s: s.lower(), reverse=True)
    return res

def get_all_programmists(data: List):
    return list(filter(lambda s: "программист" in s.lower(), data))

def add_python_expirience(data):
    return list(map(lambda job: job + " с опытом Python", data))

def add_salary(data):
    return list(zip(data, [random.randint(100_000, 200_000) for sal in
range(len(data))]))

def executor(data : Any, actions: List[callable]):
    res = data
    for a in actions:
        res = a(res)
    return res

print("\nЗадание 7")

if __name__ == "__main__":
    action = [
        get_jobs,
        unique_reverse_sort,
        get_all_programmists,
        add_python_expirience,
        add_salary]
    print('\n'.join(f'{i[0]}, {i[1]}' for i in executor(json_data, action)))
```

print_result.py

```
def print_result(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        print(func.__name__)

        if isinstance(result, list):
```

```
            # Если результат - список, выводим элементы в столбик
            for item in result:
                print(item)
        elif isinstance(result, dict):
            # Если результат - словарь, выводим ключи и значения через =
            for key, value in result.items():
                print(f'{key} = {value}')
        else:
            # Иначе просто выводим результат
            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]

print("\nЗадание 5")
print('!!!!!!!!')
test_1()
test_2()
test_3()
test_4()
```

sort.py

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

print("\nЗадание 4")
def sort():
    # Способ 1: С использованием lambda-функции
    result_with_lambda = sorted(data, key=lambda x: abs(x), reverse=True)
    print("С lambda:", result_with_lambda)
sort()
# Способ 2: Без использования lambda-функции
result = sorted(data, key=abs, reverse=True)
print("Без lambda:", result)
```

unique.py

```
from gen_random import gen_random
```

```python
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, **kwargs):
        self.items = iter(items)
        self.seen = set()
        self.ignore_case = kwargs.get('ignore_case', False)

    def __next__(self):
        while True:
            item = next(self.items)

            # Для сравнения используем ключ
            if self.ignore_case and isinstance(item, str):
                key = item.lower()
            else:
                key = item

            # Если элемент еще не встречался, добавляем его и возвращаем
            if key not in self.seen:
                self.seen.add(key)
                return item

    def __iter__(self):
        return self

print("\nЗадание 3")
print("Тест 1 - список с дубликатами:")
data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
for item in Unique(data):
    print(item, end=' ')
print()

print("\nТест 2 - генератор случайных чисел:")
data = gen_random(10, 1, 3)
for item in Unique(data):
    print(item, end=' ')
print()

print("\nТест 3 - строки, ignore_case=False:")
data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
for item in Unique(data):
    print(item, end=' ')
print()

print("\nТест 4 - строки, ignore_case=True:")
data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
for item in Unique(data, ignore_case=True):
    print(item, end=' ')
print()
```

data.json

```json
[
    {"job-name": "Frontend разработчик", "salary": 140000},
    {"job-name": "Backend разработчик", "salary": 160000},
    {"job-name": "Fullstack программист", "salary": 170000},
```

```
    {"job-name": "Data Engineer", "salary": 155000},
    {"job-name": "ML инженер", "salary": 180000},
    {"job-name": "QA инженер", "salary": 120000},
    {"job-name": "UX/UI дизайнер", "salary": 110000},
    {"job-name": "Product менеджер", "salary": 130000},
    {"job-name": "Системный администратор", "salary": 125000},
    {"job-name": "Сетевой инженер", "salary": 135000},
    {"job-name": "DevOps инженер", "salary": 165000},
    {"job-name": "Android разработчик", "salary": 145000},
    {"job-name": "iOS разработчик", "salary": 150000},
    {"job-name": "Game Developer", "salary": 155000},
    {"job-name": "Embedded программист", "salary": 140000},
    {"job-name": "Биоинформатик", "salary": 175000},
    {"job-name": "Frontend разработчик", "salary": 142000},
    {"job-name": "Backend разработчик", "salary": 158000}
]
```

Результат выполнения

```
zhuwen@boxj:~/Zhuwen-CS-Labs-2025/lab3$ python3 main.py
Задание 1
Тест 1 - один аргумент 'title':
Ковер
Диван для отдыха

Тест 2 - один аргумент 'price':
2000

Тест 3 - два аргумента 'title', 'price':
{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха'}

Задание 2
Генерируем 5 случайных чисел от 1 до 3:
2 1 3 3 2

Генерируем 10 случайных чисел от 10 до 20:
19 12 10 16 12 19 12 15 15 11

Задание 3
Тест 1 - список с дубликатами:
1 2

Тест 2 - генератор случайных чисел:
2 3 1

Тест 3 - строки, ignore_case=False:
a A b B

Тест 4 - строки, ignore_case=True:
a b

Задание 4
С lambda: [123, 100, -100, -30, 4, -4, 1, -1, 0]
Без lambda: [123, 100, -100, -30, 4, -4, 1, -1, 0]

Задание 5
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
```

```
Задание 6
Тест cm_timer_1:
time: 2.50

Тест cm_timer_2:
time: 1.52

Задание 7
Fullstack программист с опытом Python, 124929
Embedded программист с опытом Python, 149996
```