

**Московский государственный технический
университет им. Н.Э. Баумана.**
Факультет «Информатика и управление»
Кафедра «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №3
«Функциональные возможности языка Python.»

Выполнил:

студент группы ИУ5-51Б

Забелина Варвара

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

Подпись и дата

г. Москва, 2020 г.

Задание лабораторной работы

Задание лабораторной работы состоит из решения нескольких задач.

Задача 1 (файл field.py)

Необходимо реализовать генератор field. Генератор field последовательно выдает значения ключей словаря.

В качестве первого аргумента генератор принимает список словарей, дальше через *args генератор принимает неограниченное количество аргументов.

Если передан один аргумент, генератор последовательно выдает только значения полей, если значение поля равно None, то элемент пропускается.

Если передано несколько аргументов, то последовательно выдаются словари, содержащие данные элементы. Если поле равно None, то оно пропускается. Если все поля содержат значения None, то пропускается элемент целиком.

Решение:

```
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
    {'foo': 'bar'},
    {'title': None}
]

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

    if len(args) == 1:
        for obj in items:
            if args[0] in obj and obj[args[0]] is not None:
                yield obj[args[0]]
    else:
        for obj in items:
            res = {}
            for prop in args:
                if prop in obj and obj[prop] is not None:
                    res[prop] = obj[prop]
            if len(res) > 0:
                yield res

f = field(goods, 'title', 'price')

for i in f:
    print(i)
```

Результат работы:

```
lab3/lab python fp/field.py
{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха', 'price': 5300}
```

Задача 2 (файл gen_random.py)

Необходимо реализовать генератор gen_random(количество, минимум, максимум), который последовательно выдает заданное количество случайных чисел в заданном диапазоне от минимума до максимума, включая границы диапазона.

Пример: gen_random(5, 1, 3) должен выдать 5 случайных чисел в диапазоне от 1 до 3, например 2, 2, 3, 2, 1.

Решение:

```
import random

def gen_random(num_count, begin, end):
    for i in range(num_count):
        yield random.randrange(begin, end + 1)

f = gen_random(5, 1, 3)

for i in f:
    print(i)
```

Результат работы:

```
lab3/lab_python_fp/gen_random.py
2
1
1
3
2
```

Задача 3 (файл unique.py)

Необходимо реализовать итератор Unique(данные), который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.

Конструктор итератора также принимает на вход именованный bool-параметр ignore_case, в зависимости от значения которого будут считаться одинаковыми строки в разном регистре. По умолчанию этот параметр равен False.

При реализации необходимо использовать конструкцию **kwargs.

Итератор должен поддерживать работу как со списками, так и с генераторами.

Итератор не должен модифицировать возвращаемые значения.

Решение:

```
from gen_random import gen_random

class Unique:
    """Итератор, оставляющий только уникальные значения."""

    def __init__(self, data, **kwargs):
        self.used_elements = set()
        self.data = data
```

```

        self.index = 0
        self.ignore_case = False
        if 'ignore_case' in kwargs.keys():
            self.ignore_case = kwargs['ignore_case']

    def __iter__(self):
        return self

    def __next__(self):
        while True:
            if self.index >= len(self.data):
                raise StopIteration
            else:
                current = self.data[self.index]
                self.index = self.index + 1
                if self.ignore_case:
                    if current.upper() not in self.used_elements:
                        # Добавление в множество производится
                        # с помощью метода add
                        self.used_elements.add(current.upper())
                        return current
                    else:
                        if current not in self.used_elements:
                            # Добавление в множество производится
                            # с помощью метода add
                            self.used_elements.add(current)
                            return current
                else:
                    if current not in self.used_elements:
                        # Добавление в множество производится
                        # с помощью метода add
                        self.used_elements.add(current)
                        return current

def uniqueSort(arr):
    tmp = []
    for i in Unique(arr, ignore_case=True):
        tmp.append(i)
    return sorted(tmp)

for i in Unique(["ABC", "dBeRt", "dbert", "abc"], ignore_case=True):
    print(i, end=" ")
print()
f = gen_random(7, 1, 3)
for i in f:
    print(i, end=" ")
print()
for i in Unique([1, 6, 4, 3, 6, 4, 3, 2, 76, 3, 23, 4]):
    print(i, end=" ")
print()

```

Результат работы:

```

ABC dBeRt
1 3 2 2 3 2 2
1 6 4 3 2 76 23

```

Задача 4 (файл sort.py)

Дан массив 1, содержащий положительные и отрицательные числа.

Необходимо **одной строкой кода** вывести на экран массив 2, которые содержит значения массива 1, отсортированные по модулю в порядке убывания. Сортировку необходимо осуществлять с помощью функции sorted. Пример:

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

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

Необходимо решить задачу двумя способами:

1. С использованием lambda-функции.
2. Без использования lambda-функции.

Решение:

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

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

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

Результат работы:

```
lab3/lab_python_fp/sort.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
```

Задача 5 (файл print_result.py)

Необходимо реализовать декоратор print_result, который выводит на экран результат выполнения функции.

- Декоратор должен принимать на вход функцию, вызывать её, печатать в консоль имя функции и результат выполнения, после чего возвращать результат выполнения.
- Если функция вернула список (list), то значения элементов списка должны выводиться в столбик.
- Если функция вернула словарь (dict), то ключи и значения должны выводиться в столбик через знак равенства.

Решение:

```
def print_result(func):
    def wrapper(*func_args, **func_kwargs):
        print(func.__name__)
        arr = func(*func_args, **func_kwargs)
        if isinstance(arr, int):
            print(arr)
        elif isinstance(arr, str):
            print(arr)
```

```

        elif isinstance(arr, list):
            for i in arr:
                print(i)
        elif isinstance(arr, dict):
            for key, value in arr.items():
                print(key, "=", value)
        return arr
    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__':
    print('!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()

```

Результат работы:

```

lab3/lab_python_fp/print_result.py
!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2

```

Задача 6 (файл cm_timer.py)

Необходимо написать контекстные менеджеры cm_timer_1 и cm_timer_2, которые считают время работы блока кода и выводят его на экран.

Решение:

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

@contextmanager
def cm_timer_1():
    start = datetime.now()
    yield
    result = datetime.now() - start
    print(result)

@contextmanager
def cm_timer_2():
    start = time()
    yield
    print('Duration: {}'.format(time() - start))
```

Задача 7 (файл process_data.py)

- В предыдущих задачах были написаны все требуемые инструменты для работы с данными. Применим их на реальном примере.
- В файле [data_light.json](#) содержится фрагмент списка вакансий.
- Структура данных представляет собой список словарей с множеством полей: название работы, место, уровень зарплаты и т.д.
- Необходимо реализовать 4 функции - f1, f2, f3, f4. Каждая функция вызывается, принимая на вход результат работы предыдущей. За счет декоратора @print_result печатается результат, а контекстный менеджер cm_timer_1 выводит время работы цепочки функций.
- Предполагается, что функции f1, f2, f3 будут реализованы в одну строку. В реализации функции f4 может быть до 3 строк.
- Функция f1 должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна игнорировать регистр. Используйте наработки из предыдущих задач.
- Функция f2 должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”. Для фильтрации используйте функцию filter.
- Функция f3 должна модифицировать каждый элемент массива, добавив строку “с опытом Python” (все программисты должны быть знакомы с

Python). Пример: Программист С# с опытом Python. Для модификации используйте функцию map.

- Функция f4 должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей и присоединить её к названию специальности. Пример: Программист С# с опытом Python, зарплата 137287 руб. Используйте zip для обработки пары специальность — зарплата.

Решение:

```
import json
import sys
from cm_timer import cm_timer_1
from print_result import print_result
from unique import uniqueSort
from gen_random import gen_random
from time import sleep

path = "lab3\lab_python_fp\data_light.json"

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

@ print_result
def f1(arg):
    return uniqueSort([elem['job-name'] for elem in arg])

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

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

@ print_result
def f4(arg):
    return list(map(lambda x: x + ", зарплата " + str(*gen_random(1, 100000, 200000)) + " руб", arg))

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