**Московский государственный технический**

**Университет им. Н.Э. Баумана**

**Факультет «Радиотехничский»**

**Кафедра «Системы обработки информации и управления»**

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №3-4

«Функциональные возможности языка Python»

Выполнил:

студент группы РТ5-31Б

Фешин Д.А.

Проверил:

Гапанюк Е.Ю.

2024 г.

**Задание**

1 Необходимо создать виртуальное окружение и установить в него хотя бы

один внешний пакет с использованием pip.

2 Необходимо разработать программу, реализующую работу с классами.

Программа должна быть разработана в виде консольного приложения на

языке Python 3

3 Все файлы проекта (кроме основного файла main.py) должны располагаться

в пакете lab\_python\_oop.

4 Каждый из нижеперечисленных классов должен располагаться в отдельном

файле пакета lab\_python\_oop.

5 Абстрактный класс «Геометрическая фигура» содержит абстрактный метод

для вычисления площади фигуры. Подробнее про абстрактные классы и

методы Вы можете прочитать здесь.

6 Класс «Цвет фигуры» содержит свойство для описания цвета

геометрической фигуры. Подробнее про описание свойств Вы можете

прочитать здесь.

7 Класс «Прямоугольник» наследуется от класса «Геометрическая фигура».

Класс должен содержать конструктор по параметрам «ширина», «высота» и

«цвет». В конструкторе создается объект класса «Цвет фигуры» для

хранения цвета. Класс должен переопределять метод, вычисляющий

площадь фигуры.

8 Класс «Круг» создается аналогично классу «Прямоугольник», задается

параметр «радиус». Для вычисления площади используется константа

math.pi из модуля math.

9 Класс «Квадрат» наследуется от класса «Прямоугольник». Класс должен

содержать конструктор по длине стороны. Для классов «Прямоугольник»,

«Квадрат», «Круг»:

10 Определите метод "repr", который возвращает в виде строки основные

параметры фигуры, ее цвет и площадь. Используйте метод format -

https://pyformat.info/

11 Название фигуры («Прямоугольник», «Квадрат», «Круг») должно задаваться

в виде поля данных класса и возвращаться методом класса.

12 В корневом каталоге проекта создайте файл main.py для тестирования Ваших

классов (используйте следующую конструкцию -

https://docs.python.org/3/library/\_\_main\_\_.html). Создайте следующие объекты

и выведите о них информацию в консоль (N - номер Вашего варианта по

списку группы):

13 Прямоугольник синего цвета шириной N и высотой N.

14 Круг зеленого цвета радиусом N.

15 Квадрат красного цвета со стороной N.

16 Также вызовите один из методов внешнего пакета, установленного с

использованием pip.

**Текст программы**

Задача 1

def field(items, \*args):

for item in items:

result = {}

for key in args:

if key in item:

result[key] = item[key]

if result:

yield result if len(args) > 1 else list(result.values())[0]

goods = [

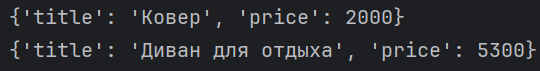
{'title': 'Ковер', 'price': 2000, 'color': 'green'},

{'title': 'Диван для отдыха', 'color': 'black', 'price': 5300}

]

for item in field(goods, 'title', 'price'):

print(item)



Задача 2

import random

def gen\_random(num\_count, begin, end):

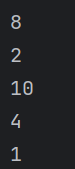
for \_ in range(num\_count):

yield random.randint(begin, end)

# Сгенерируем 5 случайных чисел от 1 до 10

for number in gen\_random(5, 1, 10):

print(number)



Задача 3

class Unique:

def \_\_init\_\_(self, items, ignore\_case=False):

self.items = iter(items)

self.found = set()

self.ignore\_case = ignore\_case

def \_\_iter\_\_(self):

return self

def \_\_next\_\_(self):

while True:

try:

item = next(self.items)

key = item.lower() if self.ignore\_case and isinstance(item, str) else item

if key not in self.found:

self.found.add(key)

return item

except StopIteration:

raise StopIteration

data = [1, 1, 1, 2, 2, 3, 'a', 'A', 'b']

# Уникальные элементы без учета регистра

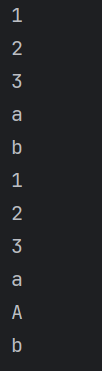
for item in Unique(data, ignore\_case=True):

print(item)

# Уникальные элементы с учетом регистра

for item in Unique(data):

print(item)



Задача 4

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

if \_\_name\_\_ == '\_\_main\_\_':

# Сортировка с использованием lambda-функции

result\_with\_lambda = sorted(data, key=lambda x: abs(x), reverse=True)

# Сортировка без использования lambda-функции

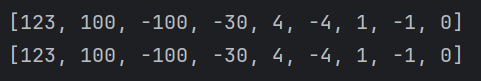
def key\_function(x):

return abs(x)

result = sorted(data, key=key\_function, reverse=True)

print(result)

print(result\_with\_lambda)



Задача 5

def print\_result(func):

def wrapper(\*args, \*\*kwargs):

result = func(\*args, \*\*kwargs)

print(func.\_\_name\_\_)

if isinstance(result, list):

for item in result:

print(item)

elif isinstance(result, dict):

for key, value in result.items():

print(f"{key} = {value}")

else:

print(result)

return result

return wrapper

@print\_result

def test\_1():

return 1

@print\_result

def test\_2():

return 'rt5'

@print\_result

def test\_3():

return {'a': 1, 'b': 2}

@print\_result

def test\_4():

return [1, 2]

if \_\_name\_\_ == '\_\_main\_\_':

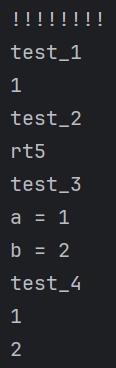
print('!!!!!!!!')

test\_1()

test\_2()

test\_3()

test\_4()



Задача 6

import time

import contextlib

def timer(func):

"""Декоратор для измерения времени выполнения функции."""

@contextlib.contextmanager

def context\_manager(\*args, \*\*kwargs):

start\_time = time.perf\_counter() # Более точное измерение времени

yield

end\_time = time.perf\_counter()

print(f"Время выполнения функции {func.\_\_name\_\_}: {end\_time - start\_time:.5f} секунд")

return context\_manager

@timer

def my\_long\_function():

# Какая-то долгая операция

for \_ in range(1000000):

pass

if \_\_name\_\_ == '\_\_main\_\_':

with my\_long\_function():

pass



Задача 7

import json

import sys

from random import randint

import time

import contextlib

def timer(func):

"""Декоратор для измерения времени выполнения функции."""

@contextlib.contextmanager

def context\_manager(\*args, \*\*kwargs):

start\_time = time.perf\_counter()

yield

end\_time = time.perf\_counter()

print(f"Время выполнения функции {func.\_\_name\_\_}: {end\_time - start\_time:.5f} секунд")

return context\_manager

def print\_result(func):

"""Декоратор для печати результата функции."""

def wrapper(\*args, \*\*kwargs):

result = func(\*args, \*\*kwargs)

print(func.\_\_name\_\_)

print(result)

return result

return wrapper

def load\_data(path):

"""Загружает данные из JSON-файла."""

with open(path) as f:

return json.load(f)

@print\_result

@timer

def process\_data(data):

# Извлечение уникальных профессий

unique\_jobs = sorted(set(item['job\_title'].lower() for item in data), key=str.casefold)

# Фильтрация программистов

programmers = list(filter(lambda x: x['job\_title'].lower().startswith('программист'), data))

# Добавление опыта Python

programmers\_with\_python = list(map(lambda x: x['job\_title'] + ' с опытом Python', programmers))

# Генерация зарплат и формирование итогового результата

salaries = (randint(100000, 200000) for \_ in programmers\_with\_python)

result = [f"{job}, зарплата {salary} руб." for job, salary in zip(programmers\_with\_python, salaries)]

return result

if \_\_name\_\_ == '\_\_main\_\_':

path = sys.argv[1]

data = load\_data(path)

process\_data(data)