Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

# ОТЧЕТ
# ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
## дисциплины «Объектно-ориентированное программирование»

Выполнил:
Кожуховский Виктор Андреевич
3 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной
техники и автоматизированных систем
», очная форма обучения

_____
(подпись)

Проверил:
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Порядок выполнения работы:

1. Изучил теоретический материал работы.

2. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

3. Выполнил клонирование созданного репозитория.

4. Дополнил файл .gitignore необходимыми правилами для работы с IDE.

5. Организовал свой репозиторий в соответствие с моделью ветвления git-flow.

6. Создал проект в папке репозитория.

7. Проработал примеры лабораторной работы.

8. Выполнил индивидуальное задание для варианта 14.

Задание 1

Парой называется класс с двумя полями, которые обычно имеют имена first и second. Требуется реализовать тип данных с помощью такого класса. Во всех заданиях обязательно должны присутствовать:

- метод инициализации __init__ ; метод должен контролировать значения аргументов на корректность;
- ввод с клавиатуры read ;
- вывод на экран display .

Реализовать внешнюю функцию с именем make_тип() , где тип — тип реализуемой структуры. Функция должна получать в качестве аргументов значения для полей структуры и возвращать структуру требуемого типа. При передаче ошибочных параметров следует выводить сообщение и заканчивать работу.

В раздел программы, начинающийся после инструкции if __name__ = '__main__': добавить код, демонстрирующий возможности разработанного класса.

Поле first — дробное положительное число, оклад; поле second — целое число, количество отработанных дней в месяце. Реализовать метод summa () —

вычисление начисленной суммы за данное количество дней для заданного месяца:

Оклад / дни месяца * отработанные дни

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

class Pair:
    def __init__(self, first=0.0, second=0):
        if not isinstance(first, (int, float)) or not isinstance(second, int):
            raise ValueError("Некорректные значения аргументов")

        self.first = float(first)
        self.second = int(second)

    def read(self, prompt=None):
        line = input() if prompt is None else input(prompt)
        parts = line.split()

        if len(parts) != 2:
            raise ValueError("Введите два значения")

        self.first = float(parts[0])
        self.second = int(parts[1])

    def display(self):
        print(f"First: {self.first}, Second: {self.second}")

    def summa(self, work_ddays):
        if work_ddays <= 0:
            raise ValueError(
                "Количество дней в месяце должно быть положительным")

        return self.first / work_ddays * self.second


def make_pair(first, second):
    try:
        return Pair(first, second)
    except ValueError as e:
        print(f"Ошибка: {e}")
        exit(1)


if __name__ == '__main__':
    p1 = make_pair(3000.250, 20)
    p1.display()
    print(f"Summa: {p1.summa(30)}")

    p2 = Pair()
    p2.read("Введите оклад и количество отработанных дней: ")
    p2.display()
    print(f"Summa: {p2.summa(30)}")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  SEARCH ERROR
PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" c:/Users/viktor/Desktop/ncfu/
First: 3000.25, Second: 20
Summa: 2000.1666666666667
Введите оклад и количество отработанных дней: 12.56 23
First: 12.56, Second: 23
Summa: 9.629333333333333
```

Рисунок 1. Код решения индивидуального задания 1 и его выполнение

Задание 2

Составить программу с использованием классов и объектов для решения задачи. Во всех заданиях, помимо указанных в задании операций, обязательно должны быть реализованы следующие методы:

- метод инициализации __init__ ;
- ввод с клавиатуры read ;
- вывод на экран display.

В раздел программы, начинающийся после инструкции if __name__ = '__main__': добавить код, демонстрирующий возможности разработанного класса.

Создать класс Payment (зарплата). В классе должны быть представлены поля: фамилия, имя, отчество, оклад, год поступления на работу, процент надбавки, подоходный налог, количество отработанных дней в месяце, количество рабочих дней в месяце, начисленная и удержанная суммы. Реализовать методы: вычисления начисленной суммы, вычисления удержанной суммы, вычисления суммы, выдаваемой на руки, вычисления стажа. Стаж вычисляется как полное количество лет, прошедших от года поступления на работу, до текущего года. Начисления представляют собой сумму, начисленную за отработанные дни, и надбавки, то есть доли от первой суммы. Удержания представляют собой отчисления в пенсионный фонд (1% от начисленной суммы) и подоходный налог. Подоходный налог составляет 13% от начисленной суммы без отчислений в пенсионный фонд.

```
        self.work_days = int(parts[7])
        self.working_days = int(parts[8])

    def display(self):
        print(
            f"Фамилия: {self.surname}, Имя: {self.name}, Отчество: {self.patronymic}")
        print(
            f"Оклад: {self.salary}, Год поступления на работу: {self.year_of_employment}")
        print(
            f"Процент надбавки: {self.bonus_percentage}, Подоходный налог: {self.income_tax}")
        print(
            f"Отработанные дни: {self.work_days}, Рабочие дни: {self.working_days}")
        print(
            f"Начисленная сумма: {self.accrued_amount}, Удержанная сумма: {self.deducted_amount}")

    def calculate_accrued_amount(self):
        base_amount = self.salary / self.working_days * self.work_days
        bonus_amount = base_amount * (self.bonus_percentage / 100)
        self.accrued_amount = base_amount + bonus_amount
        return self.accrued_amount

    def calculate_deducted_amount(self):
        pension_fund_deduction = self.accrued_amount * 0.01
        income_tax_deduction = (
            self.accrued_amount - pension_fund_deduction) * (self.income_tax / 100)
        self.deducted_amount = pension_fund_deduction + income_tax_deduction
        return self.deducted_amount

    def calculate_net_amount(self):
        return self.accrued_amount - self.deducted_amount

    def calculate_experience(self):
        current_year = datetime.now().year
        return current_year - self.year_of_employment


if __name__ == '__main__':
    p1 = Payment("Иванов", "Иван", "Иванович", 50000, 2015, 10, 13, 20, 22)
    p1.display()
    print(f"Начисленная сумма: {p1.calculate_accrued_amount()}")
    print(f"Удержанная сумма: {p1.calculate_deducted_amount()}")
    print(f"Сумма на руки: {p1.calculate_net_amount()}")
    print(f"Стаж: {p1.calculate_experience()} лет")

    p2 = Payment("", "", "", 0, 0, 0, 0, 0, 0)
    p2.read("Введите фамилию, имя, отчество, оклад, год поступления, процент надбавки, подоходный налог, отработанные дни, рабочие дни: ")
    p2.display()
    print(f"Начисленная сумма: {p2.calculate_accrued_amount()}")
    print(f"Удержанная сумма: {p2.calculate_deducted_amount()}")
    print(f"Сумма на руки: {p2.calculate_net_amount()}")
    print(f"Стаж: {p2.calculate_experience()} лет")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  SEARCH ERROR        Python

Фамилия: Иванов, Имя: Иван, Отчество: Иванович
Оклад: 50000.0, Год поступления на работу: 2015
Процент надбавки: 10.0, Подоходный налог: 13.0
Отработанные дни: 20, Рабочие дни: 22
Начисленная сумма: 0.0, Удержанная сумма: 0.0
Начисленная сумма: 49999.99999999999
Удержанная сумма: 6934.999999999999
Сумма на руки: 43064.99999999999
Стаж: 9 лет
Введите фамилию, имя, отчество, оклад, год поступления, процент надбавки, подоходный налог, отработанные дни, рабочие дни: Влад Владов Владович 230000 2001 12 45 7895 12
Фамилия: Влад, Имя: Владов, Отчество: Владович
Оклад: 230000.0, Год поступления на работу: 2001
Процент надбавки: 12.0, Подоходный налог: 45.0
Отработанные дни: 7895, Рабочие дни: 12
Начисленная сумма: 0.0, Удержанная сумма: 0.0
Начисленная сумма: 169479333.33333334
Удержанная сумма: 77197836.33333333
Сумма на руки: 92281497.00000001
```

Рисунок 2. Код решения индивидуального задания 2 и его выполнение

9. Зафиксировал сделанные изменения в репозитории.

10. Выполнил слияние ветки для разработки с веткой master/main.

11. Отправил сделанные изменения на сервер GitHub.

Контрольные вопросы:

1. Как осуществляется объявление класса в языке Python?

Классы объявляются с помощью ключевого слова class и имени класса:

```
class MyClass:
    var = ... # некоторая переменная
```

def do_smt(self):

2. Чем атрибуты класса отличаются от атрибутов экземпляра?

Атрибуты класса определены внутри класса, но вне каких-либо методов. Их значения одинаковы для всех экземпляров этого класса. Так что вы можете рассматривать их как тип значений по умолчанию для всех наших объектов.

Что касается переменных экземпляра, они хранят данные, уникальные для каждого объекта класса. В этой теме мы рассмотрим только атрибуты класса, но не волнуйтесь, у вас будет достаточно времени, чтобы узнать больше и об атрибутах экземпляра.

3. Каково назначение методов класса?

Методы класса определяют поведение объектов класса. Они могут изменять состояние объекта или выполнять действия, связанные с объектом.

4. Для чего предназначен метод __init__() класса?

Метод __init__ является конструктором. Конструкторы — это концепция объектно-ориентированного программирования. Класс может иметь один и только один конструктор. Если __init__ определен внутри класса, он автоматически вызывается при создании нового экземпляра класса.

5. Каково назначение self ?

self представляет собой ссылку на текущий экземпляр класса и используется для доступа к атрибутам и методам класса изнутри.

6. Как добавить атрибуты в класс?

Аргумент self представляет конкретный экземпляр класса и позволяет нам получить доступ к его атрибутам и методам.

7. Как осуществляется управление доступом к методам и атрибутам в языке Python?

В Python используется соглашение об именовании для управления доступом:

Приватные атрибуты и методы начинаются с одного или двух подчеркиваний (например, _private или __very_private).

Публичные атрибуты и методы не имеют подчеркиваний.

8. Каково назначение функции isinstance ?

Встроенная функция isinstance(obj, Cls) , используемая при реализации методов арифметических операций и операций отношения, позволяет узнать что некоторый объект obj является либо экземпляром класса Cls либо экземпляром одного из потомков класса Cls.