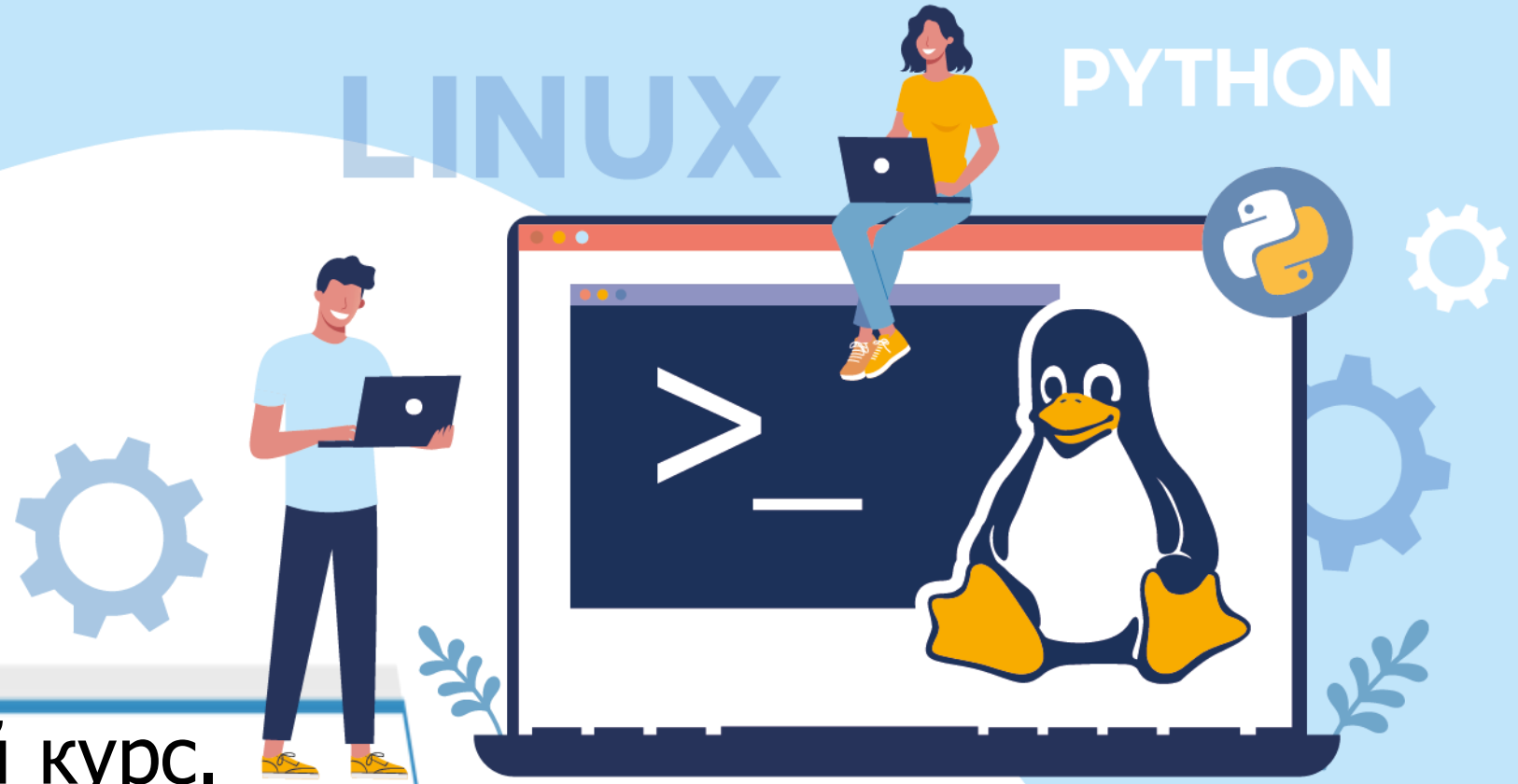
ТЕНЗОР



Python. Базовый курс.

Занятие 8.

1. ООП
2. Инкапсуляция, наследование, полиморфизм.
3. Перегрузка операторов.
4. Декораторы.

Никита Горинов

ООП

- **ООП** — объектно-ориентированное программирование. Объекты и классы лежат в основе.
- **Класс** — тип, описывающий устройство объектов.
- **Объект** — экземпляр класса.
- **Инкапсуляция** — сокрытие деталей реализации.
- **Наследование** — повторное использование компонентов.
- **Полиморфизм** — разное поведение одного и того же метода в разных классах.
- **Перегрузка операторов** — один из способов реализации полиморфизма при наследовании.

Инкапсуляция

- **Инкапсуляция** — Механизм языка, позволяющий объединить данные и методы, работающие с этими данными в единый объект, и скрыть детали реализаци и от пользователя.
- В Python отсутствует контроль сокрытия и исполнение инкапсуляции. Сокрытие в python существует на уровне соглашения между про граммистами (внутренние и внешние атрибуты).
- Одиночное подчеркивание — соглашение о приватном использовании только внутри классов. Переменная или метод не предназначены для использования вне методов класса. Атрибут доступен по имени.
- Двойное подчеркивание — то же самое, что и одиночное подчеркивание, только атрибут становится недоступным по имени, остается доступным по «`_Имя_класса__Имя_атрибута`».

ТЕНЗОР

Полиморфизм

- Полиморфизм — разное поведение одного и того же метода в разных классах.
- В Python нет возможности определить несколько методов в пределах одного класса с одинаковым именем, но различными аргументами.

Перегрузка операторов

- Перегрузка операторов — способ реализации полиморфизма, при котором в новых классах, наследующих другие классы, мы можем задать свою реализацию какого-либо метода.
- Магические методы — методы класса, специальные методы, с помощью которых вы можете добавить в Ваши классы «магию». Они всегда обрамлены двумя нижними подчеркиваниями (например, `__init__` или `__lt__`).
- https://habr.com/ru/post/186608/

Декораторы

- Декоратор — обертка, которая дает возможность изменить поведение функции, не изменяя ее код.
- Декораторы в python — это просто синтаксический сахар.
- Декораторы замедляют вызов функции.
- Нельзя «раздекорировать» функцию.
- Декораторы оборачивают функции, что затрудняет отладку. При этом мы можем использовать декораторы для упрощения отладки, например, если мы не хотим изменять код, который еще не устоялся.
- Декораторы могут быть использованы для расширения возможностей функций из сторонних библиотек, код которых мы не можем изменять.
- Полезно использовать декораторы для расширения различных функций одним и тем же кодом, без повторного его переписывания каждый раз.

Домашнее задание

1. Написать класс “animals” (животные), который включает общие признаки и поведения животных. От “animals” наследовать класс “mammals” (млекопитающие), который включает общие признаки и поведения млекопитающих. От “mammals” наследовать “human” (человек), “cat”(кот), “dog”(собака), у каждого свои признаки и поведения.

2. Написать класс “Student”, который наследует класс human, у которого среди прочих признаков есть “Количество сданных дз”.

3. * Перегрузить операторы сравнения так, чтобы студенты сравнивались по количеству сданных ими дз.

4. * Написать функцию. Для неё написать декораторы для следующего функционала:

5. ◦ логирование выполнения функции;

6. ◦ время выполнения функции;

7. ◦ замедлить выполнение кода. Ограничить частоту вызова функции.

ТЕНЗОР

Вопросы?