Слайд №2. Магические методы. В ЯП Python все является объектом. Магические методы еще называют: dunder-методы (от англ. сокращения double underscope). Они всё в объектно-ориентированном Питоне.

Магические методы, или базовые методы – способствуют реализации свойств объекта при их взаимодействие (добавляют «магию»).

Они всегда обрамлены двумя нижними подчеркиваниями (например, `__init__` или `__lt__`).

Слайд №3. Конструирование и инициализация

Слайд №4. Метод _ _init_ _ («dunder-init», «дандер инит») **–** метод отвечающий за инициализацию (задание начального значения) экземпляров класса после их создания.

Инициализация программы заключается в задании начальных значений или установке в нуль программных переменных (адресов, счетчиков, переключателей, указателей и т. п.) перед выполнением программы.

Инициализатор позволяет получить полностью настроенный экземпляр класса.

Слайд №6. Метод _ _del_ _ - деструктор класса. Деструктор – это функция, которая вызывается автоматически при выхода объекта из области действия или которая явно уничтожена. Этот метод может быть довольно удобным для объектов, которые могут требовать дополнительных чисток во время удаления.

Слайд №7. Представление своих классов. Часто бывает полезно представление класса в виде строки. В ЯП Питоне существует несколько методов, которые вы можете определить для настройки поведения встроенных функций при представлении вашего класса.

Слайд №8. Метод _ _str_ _ – магический метод для отображения информации об объекте класса для пользователей. Данный метод должен вывести красивое, читабельное информационное сообщение, отражающее объект. `__str__()` имеет смысл определять для объектов, для которых существует «естественное» человеко-читаемое (неспецифичное для Питона) представление.

Слайд №9. Метод _ _repr_ _ .`__repr__` - это однозначное представление объекта в виде строки, которое можно использовать, чтобы воссоздать точно такой же объект, а если это невозможно, то вывести *какое-нибудь полезное сообщение*. Машино-читаемая форма.

Слайд №10.`__len__(self)` Возвращает количество элементов в контейнере. Часть протоколов для изменяемого и неизменяемого контейнеров.

Слайд №11.__abs__(self) - позволяет применять функцию abs() к экземплярам класса.

Слайд №12.

- __add__() – для операции сложения(+);**Слайд №13.**
- __sub__() – для операции вычитания(-);

- __mul__() – для операции умножения(*);**Слайд №14.**
- __truediv__() – для операции деления(/).
- __floordiv__(self, other) – для операции //;
- __mod__(self, other) – для операции %.

Слайд №15

- **__eq__() – для равенства == Слайд №16**
- **__ne__() – для неравенства !=**
- **__lt__() – для оператора меньше <**
- **__le__() – для оператора меньше или равно <=**
- **__gt__() – для оператора больше >**
- **__ge__() – для оператора больше или равно >=**

Слайд №17. В стандартном поведении она возвращает True для непустых объектов и False – для пустых. В действительности, эта функция всегда возвращает True для любых объектов пользовательского класса.

Слайд №18-20. Здесь речь пойдет о следующем их наборе:

- __getitem__(self, item) – получение значения по ключу item;
- __setitem__(self, key, value) – запись значения value по ключу key;
- __delitem__(self, key) – удаление элемента по ключу key.