УРОК 1

ТЕМЫ:

A. Суть ООП
B. Создание первых классов
C. Атрибуты и Методы классов
D. Наследование

ДЕТАЛИ:

- Что такое класс и что такое объект
- Пример класса Car
- Атрибуты объекта
- Конструктор (__init__(self))
- Значения по умолчанию в конструкторе
- Создание объектов (аргументы с названием атрибута и без)
- Адрес объекта - self
- Методы
- Общие атрибуты класса
- Основные принципы ООП - Наследование
- Родительский и дочерний класс
- Связка по конструкторам
- Переопределение атрибутов уровня класса
- Пример Transport->Plane, Car->Truck

ДЗ*:

1. Создать класс Person с атрибутами fullname, age, is_married
2. Добавить в класс Person метод introduce_myself, который бы распечатывал всю информацию о человеке
3. Создать класс Student наследовать его от класса Person и дополнить его атрибутом marks, который был бы словарем, где ключ это название урока, а значение - оценка.
4. Добавить метод в класс Student, который бы подсчитывал среднюю оценку ученика по всем предметам
5. Создать класс Teacher и наследовать его от класса Person, дополнить атрибутом experience.
6. Добавить в класс Teacher атрибут уровня класса base_salary
7. Также добавить метод в класс Teacher, который бы считал зарплату по следующей формуле: к стандартной зарплате прибавляется бонус 5% за каждый год опыта свыше 3-х лет.
8. Создать объект учителя и распечатать всю информацию о нем и высчитать зарплату
9. Написать функцию create_students, в которой создается 3 объекта ученика, эти ученики добавляются в список и список возвращается функцией как результат.

10. Вызвать функцию create_students и через цикл распечатать всю информацию о каждом ученике с его оценками по каждому предмету. Также рассчитать его среднюю оценку по всем предметам.

УРОК 2

ТЕМЫ:

A. Основные принципы ООП
B. Инкапсуляция
C. Полиморфизм

ДЕТАЛИ:

- Создание геттеров и сеттеров, аннотации @property и имя_свойства_геттера.setter
- Сокрытие методов
- Модификаторы доступа public, private
- Переопределение методов
- Добавление атрибута стороннего класса
- Пример классов Animal + Address -> Dog, Cat, Fish
- Вызов методов полиморфно

ДЗ**:

1. Создать класс Figure (фигура) с атрибутом уровня класса unit (единица измерения величин) и присвоить ему значение cm (сантиметры) или mm (миллиметры)
2. Создать приватный атрибут perimeter в классе Figure, который бы по умолчанию в конструкторе присваивался к нулю.
3. В конструкторе класса Figure должен быть только 1 входящий параметр self.
4. Создать в классе Figure геттер и сеттер для атрибута perimeter.
5. Добавить в класс Figure нереализованный публичный метод calculate_area (подсчет площади фигуры)
6. Добавить в класс Figure нереализованный публичный метод calculate_perimeter (подсчет периметра фигуры)
7. Добавить в класс Figure нереализованный публичный метод info (вывод полной информации о фигуре)
8. Создать класс Square (квадрат), наследовать его от класса Figure.
9. Добавить в класс Square атрибут side_length (длина одной стороны квадрата), атрибут должен быть приватным.
10. В конструкторе класса Square должен высчитываться периметр квадрата, посредством вызова метода calculate_perimeter и возвращаемый результат метода должен задаваться атрибуту perimeter.
11. В классе Square переопределить метод calculate_area, который бы считал и возвращал площадь квадрата.
12. В классе Square переопределить метод calculate_perimeter, который бы считал и возвращал периметр квадрата.

13. В классе Square переопределить метод info, который бы распечатывал всю информацию о квадрате следующим образом:

Например - Square side length: 5cm, perimeter: 20cm, area: 25cm.

14. Создать класс Rectangle (прямоугольник), наследовать его от класса Figure.
15. Добавить в класс Rectangle атрибут length (длина) и width (ширина), атрибуты должны быть приватными.
16. В конструкторе класса Rectangle должен высчитываться периметр прямоугольника, посредством вызова метода calculate_perimeter и возвращаемый результат метода должен задаваться атрибуту perimeter.
17. В классе Rectangle переопределить метод calculate_area, который бы считал и возвращал площадь прямоугольника.
18. В классе Rectangle переопределить метод calculate_perimeter, который бы считал и возвращал периметр прямоугольника.
19. В классе Rectangle переопределить метод info, который бы распечатывал всю информацию о прямоугольнике следующим образом:

Например - Rectangle length: 5cm, width: 8cm, perimeter: 26cm, area: 40cm.

20. В исполняемом файле создать список из 2-х разных квадратов и 3-х разных прямоугольников
21. Затем через цикл вызвать у всех объектов списка метод info

ДЗ*:
1. Создать класс Figure (фигура) с атрибутом уровня класса unit (единица измерения величин) и присвоить ему значение cm (сантиметры) или mm (миллиметры)
2. В конструкторе класса Figure должен быть только 1 входящий параметр self, то есть не должно быть атрибутов уровня объекта.
3. Добавить в класс Figure нереализованный публичный метод calculate_area (подсчет площади фигуры)
4. Добавить в класс Figure нереализованный публичный метод info(вывод полной информации о фигуре)
5. Создать класс Circle (круг), наследовать его от класса Figure.
6. Добавить в класс Circle атрибут radius (радиус круга), атрибут должен быть приватным.
7. В классе Circle переопределить метод calculate_area, который бы считал и возвращал площадь круга.
8. В классе Circle переопределить метод info, который бы распечатывал всю информацию о круге следующим образом:

Например - Circle radius: 2cm, area: 12.57cm.

9. Создать класс RightTriangle (правильный треугольник - 90 градусов), наследовать его от класса Figure.
10. Добавить в класс RightTriangle атрибут side_a (сторона а) и side_b (сторона б), атрибуты должны быть приватными.

11. В классе RightTriangle переопределить метод calculate_area, который бы считал и возвращал площадь треугольника.
12. В классе RightTriangle переопределить метод info, который бы распечатывал всю информацию о треугольнике следующим образом:
    Например - RightTriangle side a: 5cm, side b: 8cm, area: 20cm.
13. В исполняемом файле создать список из 2-х разных кругов и 3-х разных треугольников
14. Затем через цикл вызвать у всех объектов списка метод info

УРОК 3

ТЕМЫ:

A. Множественное наследование
B. Магические методы в классах

ДЕТАЛИ:

- Множественное наследование - Ромбовидное наследование (Пример с HybridCar)
- (MRO(), __mro__, help())
- Миксины (PlayMusic)
- Методы класса @classmethodСтатические методы класса
-
- Магические методы в классах https://habr.com/ru/post/186608/

ДЗ*:
1. Создать класс Computer (компьютер) с приватными атрибутами cpu и memory.
2. Добавить сеттеры и геттеры к существующим атрибутам.
3. Добавить в класс Computer метод make_computations, в котором бы выполнялись арифметические вычисления с атрибутами объекта cpu и memory.
4. Создать класс Phone (телефон) с приватным полем sim_cards_list (список симкард)
5. Добавить сеттеры и геттеры к существующему атрибуту.
6. Добавить в класс Phone метод call с входящим параметром sim_card_number и call_to_number, в котором бы распечатывалась симуляция звонка в зависимости от переданного номера сим-карты (например: если при вызове метода передать число 1 и номер телефона, распечатывается текст “Идет звонок на номер +996 777 99 88 11” с сим-карты-1 - Beeline).
7. Создать класс SmartPhone и наследовать его от 2-х классов Computer и Phone.
8. Добавить метод в класс SmartPhone use_gps с входящим параметром location, который бы распечатывал симуляцию проложения маршрута до локации.
9. В каждом классе переопределить магический метод __str__ которые бы возвращали полную информацию об объекте.
10. Перезаписать все магические методы сравнения в классе Computer (6 шт.), для того чтоб можно было сравнивать между собой объекты, по атрибуту memory.
11. Создать 1 объект компьютера, 1 объект телефона и 2 объекта смартфона

12. Распечатать информацию о созданных объектах
13. Опробовать все возможные методы каждого объекта (например: use_gps, make_computations, call, а также магические методы)

УРОК 4

ТЕМЫ:

A. Практическое закрепление пройденного материала по ООП
B. Написание RPG игры в ООП стиле

ДЕТАЛИ:

- -

ДЗ:

ОСНОВНОЕ: Добавить в проект уникальную реализацию суперспособности нижеперечисленных героев:

1. Magic должен увеличивать атаку каждого героя после каждого раунда на n-ное количество
2. Thor, удар по боссу имеет шанс оглушить босса на 1 раунд, вследствие чего босс пропускает 1 раунд и не наносит урон героям
3. Witcher, не наносит урон боссу, но получает урон от босса. Имеет 1 шанс оживить первого погибшего героя, отдав ему свою жизнь, при этом погибает сам.

ДОПОЛНИТЕЛЬНОЕ: Добавить в проект уникальную реализацию суперспособности 3-х героев на ваш выбор из списка ниже:

4. Golem, который имеет увеличенную жизнь но слабый удар. Может принимать на себя 1/5 часть урона исходящего от босса по другим игрокам
5. Avrora, которая может входить в режим невидимости на 2 раунда (т.е не получает урон от босса), в тоже время полученный урон в режиме невидимости возвращает боссу в последующих раундах. Она может исчезать только один раз за игру
6. Druid, который имеет способность рандомно призывать помощника ангела героям или же ворона боссу на 1 раунд за всю игру. "Ангел" увеличивает способность медика лечить героев на n кол-во. А ворон прибавляет агрессию (увеличивается урон на 50%), боссу если его жизнь менее 50%.
7. Hacker, который будет через раунд забирать у Босса N-ое количество здоровья и переводить его одному из героев
8. Tricky, способность которого будет состоять в том, чтобы притвориться мертвым в определенном раунде(из случайного выбора), но в следующем раунде он снова вступает в бой. При этом он не получает урон и не бьет босса когда притворился мертвым

9. AntMan, в каждом раунде он может увеличиться или же уменьшится на N-ный размер, также увеличиваются/уменьшаются жизнь и урон, после раунда он возвращается в исходный размер
10 Deku (сила удара может меняться каждый раунд с шансом 50 на 50, может усилится на 20%, 50%, 100%, но при усилении теряется хп (чем сильнее усиление, тем больше хп потеряет герой)
11. Герой Kamikadze без урона но хорошое здоровье, его способность жертвовать собой. Но он должен попасть точно в цель, иначе нанесет урон только на 50% из своего остатка жизни.
12. Герой Samurai кидает сюрикенами которые делятся на два вида: 1) Вирус наносит N-е кол-во урона. 2) Вакцина лечит на N-е кол-во единиц здоровье босса. сюрикены выбирает Рандом.
13 Герой Bomber, когда босс убивает героя он взрывается и наносит боссу дополнительный урон в 100 единиц.
14. Reaper(Жнец) - при уровне здоровья менее 30% увеличивается урон вдвое, а при 15% втрое
15. Spitfire - каждый раз когда босс убивает одного или нескольких героев то наш герой показывает агрессию на 80 единиц урона
16. Герой King, не наносить урон, только получает, с 10% шансом он может призвать героя Saitama который убьет босса с 1 удара

УРОК 5

ТЕМЫ:

Модули в python
GitLab

Виртуальная среда

ДЕТАЛИ:

- Встроенные модули Python (random, datetime, os)
- Import with alias
- Определение собственных модулей
- __name__ , __main__
- Внешние модули
- Pypi.org - termcolor, emoji
- Pip install package_name==2.2
- Работа с файлом настроек (decouple)
- pip freeze vs. pip list
- Создание и активация виртуальной среды
- Git - https://git-scm.com/downloads

ДЗ*:
1. Установить в свою виртуальную среду проекта внешний модуль python-decouple
2. В файле requirements.txt зафиксировать зависимости проекта с помощью команды pip freeze
3. Создать многомодульную игру Казино
4. Сам запуск игры в отдельном файле
5. Логика выигрыша или проигрыша в отдельном файле
Правила игры такие :
A. Есть список (list) из чисел от 1 до 30, каждый раз вы делаете ставку на определенную слоту из чисел и ставите деньги
B. Рандомно выбирается выигрышная слота, если вы выигрываете, вам причисляется удвоенная сумма, той которую вы поставили, если вы загадали не выигрышную слоту - теряете поставленную сумму
C. В начале игры у вас также есть деньги например 1000$, но в конце мы понимаем вы в выигрыше или в проигрыше
D. Значение переменной начального капитала должно считываться с системной переменной под названием MY_MONEY из файла settings.ini
E. После каждой ставки вам задается вопрос хотите ли вы сыграть еще, если да - то делаете ставку, если нет - то подводится итог игры

УРОК 6

ТЕМЫ:

A. Алгоритмы и структуры данных
B. Big O нотация

ДЕТАЛИ:

- Оценка временной сложности алгоритмов (Big O)
- Блок-схемы
- Алгоритмы поиска (Линейный и Бинарный)
- Простейшие алгоритмы сортировки (Пузырьковая сортировка и Сортировка выбором)

ДЗ*:
1. Написать функцию bubble_sort или selection_sort, принимающую в качестве входящего параметра не отсортированный список.
2. Алгоритм функции должен сортировать список методом пузырьковой сортировки или методом сортировки выбором.
3. Функция в итоге должна возвращать отсортированный список. Применить 1 раз данную функцию

4. Написать функцию binary_search, принимающую в качестве входящего параметра элемент для поиска и список в котором необходимо искать.
5. Алгоритм должен искать с помощью двоичного поиска, изображенного на блок-схеме презентации.
6. Функция в итоге должна распечатать результат. Применить 1 раз эту функцию

УРОК 7

ТЕМЫ:

A. Базы данных и СУБД
B. Работа с БД в Python

ДЕТАЛИ:

- Создание БД
- Создание таблицы и типы данных в БД
- Добавление записей в таблицу
- Изменение записей в таблице
- Удаление записей из таблицы
- Выборка данных из таблицы (WHERE -> LIKE, BETWEEN, IN, IS NULL)
- Логические операторы OR, AND, NOT
- Выражение DISTINCT и функция - strftime('%d %m %Y', '2015-12-01')
- Условная конструкция CASE
- Сортировка результатов выборки

ДЗ*:
1. Создать базу данных hw.db в sqlite через код python, используя модуль sqlite3
2. В БД создать таблицу products
3. В таблицу добавить поле id - первичный ключ тип данных числовой и поддерживающий авто-инкрементацию.
4. Добавить поле product_title текстового типа данных максимальной длиной 200 символов, поле не должно быть пустым (NOT NULL)
5. Добавить поле price не целочисленного типа данных размером 10 цифр из которых 2 цифры после плавающей точки, поле не должно быть пустым (NOT NULL) значением по-умолчанию поля должно быть 0.0
6. Добавить поле quantity целочисленного типа данных, поле не должно быть пустым (NOT NULL) значением по-умолчанию поля должно быть 0
7. Добавить функцию, которая бы добавляла в БД 15 различных товаров
8. Добавить функцию, которая меняет количество товара по id
9. Добавить функцию, которая меняет цену товара по id
10. Добавить функцию, которая удаляет товар по id

11. Добавить функцию, которая бы выбирала все товары из БД и распечатывала бы их в консоли
12. Добавить функцию, которая бы выбирала из БД товары, которые дешевле 100 сомов и количество которых больше чем 5 и распечатывала бы их в консоли
13. Добавить функцию, которая бы искала в БД товары по названию (Например: искомое слово “мыло”, должны соответствовать поиску товары с названием - “Жидкое мыло с запахом ванили”, “Мыло детское” и тд.)
14. Протестировать каждую написанную функцию

УРОК 8

ТЕМЫ:

A. Реляционные базы данных
B. Агрегационные функции и группировка данных
C. Вложенные запросы

ДЕТАЛИ:

- Реляционные базы данных (Foreign Key)
- Соединения таблиц (JOINS, UNION ALL) https://shra.ru/2017/09/sql-join-v-primerakh-s-opisaniem/
- Типы соотношений таблиц (one-to-one, one-to-many, many-to-one, many-to-many)
- Группировка данных и агрегационные функции
- Выполнение подзапросов

ДЗ*:
1. Создать таблицу countries (страны) с колонками id первичный ключ автоинкрементируемый и колонка title с текстовым не пустым названием страны.
2. Добавить 3 записи в таблицу countries
3. Добавить таблицу cities (города) с колонками id первичный ключ автоинкрементируемый, колонка title с текстовым не пустым названием города и колонка area площадь города не целочисленного типа данных со значением по умолчанием 0, а также колонка country_id с внешним ключом на таблицу countries.
4. Добавить 7 городов различных стран
5. Создать таблицу employees (сотрудники) с колонками id первичный ключ автоинкрементируемый, колонка first_name (имя) с текстовым не пустым значением, колонка last_name (фамилия) с текстовым не пустым значением, а также колонка city_id с внешним ключом на таблицу cities.
6. Добавить 15 сотрудников проживающих в разных городах.
В пунктах с 1-го по 6-й можно использовать любой вариант для работы в СУБД SqlLite (Из кода в Python или SQL запросами или через любую программу с графическим интерфейсом для управления БД).

7. Написать программу в Python, которая при запуске бы отображала фразу “Вы можете отобразить список сотрудников по выбранному id города из перечня городов ниже, для выхода из программы введите 0:”
8. Ниже фразы программа должна распечатывать список городов из вашей базы данных следующим образом

1. Бишкек
2. Ош
3. Берлин
4. Пекин
5. и тд…

9. После ввода определенного id города программа должна найти всех сотрудников из вашей базы данных проживающих в городе выбранного пользователем и отобразить информацию о них в консоли (Имя, фамилия, страна, город проживания, площадь города)