Студент группы ИС-25 Неговора Н.М.

Практическое занятие № 16

Тема: составление программ с использованием ООП.

Цель: закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составление программ с ООП в IDE PyCharm Community

Постановка задачи:

1. Создайте класс «Календарь», который имеет атрибуты год, месяц и день. Добавьте методы для определения дня недели, проверки на високосный год и определения количества дней в месяце
2. Создайте базовый класс "Животное" со свойствами "вид", "количество лап", "цвет шерсти". От этого класса унаследуйте класс "Собака" и добавьте в него свойства "кличка" и "порода".
3. Для задачи из блока 1 создать две функции, save_def и load_def, которые позволяют сохранять информацию из экземпляров класса в файл и загружать ее обратно. Использовать модуль pickle для сериализации и десериализации объектов Python в бинарном формате.

Тип Алгоритма: Линейный

Текст программы 1:

```
import calendar
import re

class CALENDAR():
    def __init__(self, day = 15, month = 10, year = 2006) -> None:
        self.year = year
        self.month = month
        self.day = day
        self.day_name = {
            0: "Понедельник",
            1: "Вторник",
            2: "Среда",
            3: "Четверг",
            4: "Пятница",
            5: "Суббота",
            6: "Воскресенье"
        }

    def get_day_of_week(self):
        """
        Возвращает день недели на русском языке
        """
        day = calendar.weekday(self.year, self.month, self.day)
        day = self.day_name[day]
        return day

    def check_for_leap_year(self):
        """
        Возвращает строку "Весокосный" или "Не весокосный", по отношению к году,
указаному в классе
        """
```

```
        result = calendar.isleap(self.year)
        if result: return "Весокосный"
        else: return "Не весокосный"

    def get_umber_of_days_in_a_month(self):
        """
        Возвращает строку "Дней в месяце: {data}", где date - количество дней в месяце
        """
        data = calendar.monthrange(self.year, self.month)[1]
        return f'Дней в месяце: {data}"


if __name__ == "__main__":
    cal = CALENDAR()
    print(cal.get_day_of_week())
    print(cal.check_for_leap_year())
    print(cal.get_umber_of_days_in_a_month())
```

Протокол работы программы 1:

```
Воскресенье
Не весокосный
Дней в месяце: 31
```

Текст программы 2:

```
class ANIMAL():
    def __init__(self, kind = "Животное", number_of_paws = 4, color = "Белый"):
        self.kind = kind
        self.number_of_paws = number_of_paws
        self.color = color

class DOG(ANIMAL):
    def __init__(self, kind="Собака", number_of_paws=4, color="Белый", name="Боб",
breed="Дворняга"):
        super().__init__(kind, number_of_paws, color)
        self.name = name
        self.breed = breed

    def get_all_attribute(self) -> list[str]:
        list_attr = [attr for attr in self.__dict__ if not callable(getattr(self, attr)) and not
attr.startswith("__")]
        list_attr = [f"  {item}: {self.__getattribute__(item)}" for item in list_attr]
        return list_attr

if __name__ == "__main__":
    bob = DOG(name="Боб", breed="Немецкая Гончая")
    bob_description = bob.get_all_attribute()
```

```
    print("\nОписание Боба:")
    print(*bob_description, sep="\n")
    print("\n")
```

Протокол работы программы 2:

```
Описание Боба:
  kind: Собака
  number_of_paws: 4
  color: Белый
  name: Боб
  breed: Немецкая Гончая
```

Текст программы 3:

```
import pickle

from task_1 import CALENDAR as OLD_CALENDAR
from task_2 import DOG as OLD_DOG

class CALENDAR(OLD_CALENDAR):
    def __init__(self, day=15, month=10, year=2006):
        super().__init__(day, month, year)

class DOG(OLD_DOG):
    def __init__(self, kind="Собака", number_of_paws=4, color="Белый", name="Боб",
breed="Дворняга"):
        super().__init__(kind, number_of_paws, color, name, breed)

def save_def(obj: object):
    with open(f"{obj.__class__.__name__}.pickle", "wb") as file:
        pickle.dump(obj, file)

def load_def(class_name):
    with open(f"{class_name}.pickle", "rb") as file:
        result = pickle.load(file)
    return result

if __name__ == "__main__":
    save_def(DOG(name="Игорь"))
    igor = load_def("DOG")
    igor_description = igor.get_all_attribute()
    print(*igor_description, sep="\n")
```

Протокол работы программы 3:

kind: Собака
number_of_paws: 4
color: Белый
name: Игорь
breed: Дворняга

Вывод: В данной практической работе Я закрепил усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрел навыки составление программ с ООП в IDE PyCharm Community