Методические указания

Урок 17.1. Написание модулей

Задачи урока:

- Написание модулей

0. Подготовка к уроку

До начала урока преподавателю необходимо:

1) Просмотреть, как ученики справились с домашним заданием
2) Прочитать методичку

1. Написание модулей

Учитель: Давайте продолжим работу с модулями. В прошлый раз мы с вами создали свой модуль calc.py и импортировали его в свою программу. Но может ли модуль работать как самостоятельная программа без импортирования? Да может. Но если мы пропишем вызов функций в нашем модуле или какие то выводы в консоль, то при импорте они выполнятся тоже, так как при импорте по факту мы запускаем весь код модуля.
Для решения данной проблемы существует условие

```
if __name__ == '__main__':
    pass
```

Разберем как работает данное условие, а также что такое __name__ и __main__
На время добавим пару строк в файл нашей основной программы и запустим

```
import calc
print(__name__)
print(calc.__name__)
```

Программа выведет __main__ и calc. Получается при выводе __name__, если программа выполняется как самостоятельная, т.е мы ее запустили, то ее имя __main__, а для импортированного модуля именем является название файла.

Зная данную информацию мы можем разобрать условие. Если имя программы __main__, то значит она запущена как основная и мы можем в блоке условия описать необходимый код. Давайте модернизируем файл calc.py: добавим небольшое меню для выбора необходимой функции.

```
def add(num1: float, num2: float) -> float:
    return num1 + num2

def sub(num1: float, num2: float) -> float:
    return num1 - num2

def mul(num1: float, num2: float) -> float:
    return num1 * num2

def div(num1: float, num2: float) -> float:
    if num2 == 0:
        return 'Деление на 0'
    return num1 / num2


if __name__ == '__main__':
    print('Я запущен как самостоятельная программа')
    choice = int(input('Выберите необходимое действие 1: +, 2: -, 3: *, 4:
/\n'))
    num1 = int(input('Введите первое число: '))
    num2 = int(input('Введите второе число: '))
    match choice:
        case 1:
            print(add(num1, num2))
        case 2:
            print(sub(num1, num2))
        case 3:
            print(mul(num1, num2))
        case 4:
            print(div(num1, num2))
        case _:
            print('Неверный выбор')
```

Теперь при запуске файла calc у нас будет выполняться код в блоке условия. Конечно функционал у нас дублирует основной файл, но в целях примера этого достаточно.

Таким образом мы можем создавать программы, а при желании импортировать из программы отдельные модули. Давайте добавим данное условие тоже в основной файл

```
from calc import add, sub, mul, div

class Calculator:
    def __init__(self) -> None:
        self.main()
```

```
    def main(self):
        print('Это калькулятор')
        while True:
            num1 = int(input('Введите первое число: '))
            num2 = int(input('Введите второе число: '))
            choice = int(input('Выберите необходимое действие 1: +, 2: -,
3: *, 4: /, 0: Выход\n'))
            match choice:
                case 0:
                    print('Для завершения нажмите Enter')
                    input()
                    break
                case 1:
                    print(add(num1, num2))
                case 2:
                    print(sub(num1, num2))
                case 3:
                    print(mul(num1, num2))
                case 4:
                    print(div(num1, num2))
                case _:
                    print('Неверный выбор')

if __name__ == '__main__':
    obj = Calculator()
```

Атрибуты модулей:

__name__	Полное имя модуля
__doc__	Строка документации
__dict__	Словарь модуля
__file__	Файл, в котором модуль определяется
__package__	Имя содержащегося пакета (если есть)
__path__	Список подкаталогов для поиска подмодулей пакета
__annotations__	Аннотации типов уровня модуля

При желании мы можем объединять модули в пакеты и работать уже с пакетами.
Пакет — это набор модулей, сгруппированных под общим именем верхнего уровня. Такое упорядочивание помогает разрешить конфликты между именами модулей в разных приложениях и отделить ваш код от стороннего. Для создания пакета требуется создать каталог и поместите в него файл __init__.py. После этого можно разместить в этом каталоге нужные файлы Python и подпакеты.
Пакет может иметь следующую структуру:

my_package/
  __init__.py
  primitive/
    __init__.py

text.py
line.py
formats/
__init__.py
jpeg.py
png.py

Команда import используется для загрузки модулей из пакетов так же, как и для простых модулей, не считая более длинных имен:

```
# Полный путь
import my_package.primitive.fill

my_package.primitive.fill.функция()

# Загрузка конкретного подмодуля
from my_package.primitive import fill

fill.функция()

# Загрузка конкретной функции из подмодуля
from my_package.primitive.fill import функция

функция()
```

Когда какая-то часть пакета импортируется впервые, сначала выполняется код из файла __init__.py (если он есть).

При импорте глубоко вложенного подмодуля выполняются все файлы __init__.py, встречающиеся при обходе структуры каталогов. Так, команда import my_package.primitive.fill сначала выполнит файл __init__.py из каталога my_package/, а затем __init__.py из primitive/.

Одна из возможных проблем при работе с пакетами связана с взаимодействием между файлом __init__.py и подмодулями. Для лучшего управления организационной сложностью подмодули пакета часто объявляют явный список экспорта, определяя переменную __all__.

```
__all__ = ['MyClass']
class MyClass:
    pass
```

Далее соответствующий файл __init__.py импортирует свои подмодули с использованием синтаксиса *:

```
__all__ = test.__all__
```

либо:

```
__all__ = [
*graph2d.__all__,
*graph3d.__all__
]
```

Помните, что вы (или другие люди) будут импортировать ваш модуль и использовать его. Модуль нельзя именовать также, как и ключевое слово. Также имена модулей нельзя начинать с цифры. И не стоит называть модуль также, как какую-либо из встроенных функций. То есть, конечно, можно, но это создаст большие неудобства при его последующем использовании.

Куда еще можно сохранить собственный модуль?Туда, где его потом можно будет найти. Пути поиска модулей указаны в переменной sys.path. В него включены текущая директория (то есть модуль можно оставить в папке с основной программой), а также директории, в которых установлен python. Кроме того, переменную sys.path можно изменять вручную, что позволяет положить модуль в любое удобное для вас место (главное, не забыть в главной программе модифицировать sys.path).

```
import sys
sys.path.append('Моя директория/')
print(sys.path)
```

Учитель: Модули свои мы создавать научились, импортировать стандартные тоже, но как же работать со сторонними, а вернее откуда их брать. Для этого при установке Python у нас с вами устанавливается pip.
Pip - Система управления пакетами, которая используется для установки и управления программными пакетами, написанными на Python.

С помощью pip мы можем устанавливать огромное количество модулей, как маленьких так и больших, например для создания серверной части сайта, телеграмм ботов, парсинга данных, создания 2D игр, работы с запросами и базами, искусственным интеллектом и многое другое. Все это обилие модулей расположено на ресурсе PyPI.

Pip позволяет с помощью одной команды установить, удалить или обновить необходимый модуль.
Предположим мы хотим установить библиотеку для создания серверной части сайта(django)

```
pip install django
```

а если удалить

```
pip uninstall django
```

2. Решение задач

Задача 1

При заданном целом числе n посчитайте n + nn + nnn.

```
def func(n):
    n1 = n
    n2 = int(str(n) * 2)
    n3 = int(str(n) * 3)
    print(n1 + n2 + n3)
func(5)
```

Дополнительно

Если на уроке остается время, то ученикам можно предложить начать прорешивать домашнее задание.

Домашняя работа

Задача 1
Есть список:
a = [1, 2, 3, 3, 4, 5]
Требуется написать функцию в отдельном модуле, которая проверяет все ли числа в данном списке, являются уникальными. Разделить функцию и ее вызов по отдельным файлам, с соответствующим импортом функции в основной файл.

```
def all_unique(numbers):
```

```
    return len(numbers) == len(set(numbers))
```