Основы языка Python. Интерактивный курс

# Модули. Определение. Применение. Подключение

# План

- Определение модуля.
- Зачем нужны модули?
- Разновидности модулей.
- Подключение модуля.

# Определение модуля

- Python позволяет поместить классы, функции, данные а также скрипты в отдельный файл и использовать их в других программах.

- Модулем в Python называется любой файл с программой.

# Зачем нужны модули

- Повторное использование кода.
- Управление пространством имен.
- Деление большого проекта на мелкие части.

# Разновидности модулей

- встроенные (math, random, ...)
- сторонние (django, PyQt5, ...)
- свои (любой .py файл)
- между своими и сторонними модулями нет принципиальной разницы, отличие только в авторе

# Варианты подключения

- модуль целиком import math
- псевдоним для модуля import math as mt
- импорт всего содержания from math import * (не рекомендуется)
- импорт конкретных функций, классов, … from math import sin, cos

Python

# Стандартные модули math, random

# План

- Библиотека math.
- Библиотека random.
- Примеры применения.

# Использование math

- Математические функции.
- Работа с числами.

# Основные функции math

- factorial - факториал числа
- exp - экспонента
- log, log2, log10 - логарифмы
- sqrt - квадратный корень
- sin, cos, asin, asoc, ...
- и многие другие

# Использование random

- Генерация случайных чисел.
- Букв.
- Элементов последовательности.

# Основные функции random

- randint - целое случайное число от A до B
- choice - случайный элемент последовательности
- shuffle - перемешивает последовательность
- random - случайное число от 0 до 1
- sample - список длиной k из последовательности
- и многие другие

Python

# Создание собственных модулей

# План

- Создание своего модуля
- Импорт данных из своего модуля
- Импорт скриптов
- if __name__ == ‘__main__’

# Импорт из своих модулей

- Совершается так же, как и из стандартных модулей
- При импорте нужно учитывать путь до модуля
- import firstmodule
- import folder.secondmodule

# Модули со скриптами

- При любом варианте импорта скрипты будут выполняться
- Если не указано никаких условий (if __name__ == ‘__main__’)
- Это обязательно нужно учитывать при импорте

# if __name__ == ‘__main__’

- Ограничивает выполнение скриптов
- При импорте код не будет выполняться
- Он будет выполняться при запуске модуля

Python

# Пакеты

# План

- Определение пакета.
- Создание пакета.
- Назначение пакета.
- Импорт из пакета.

# Определение пакета

- Пакет - каталог, включающий в себя другие пакеты и модули
- Пакет содержит файл __init__.py

# Назначение пакета

- Формирование пространства имен.
- Работа с модулями с указанием уровня вложенности.
- пакет1.пакет2.модуль.
- Уровни отделяются точкой.

# Варианты импортов

- import .модуль - внутри пакета из одного модуля в другой
- import пакет.модуль - стандартно
- import, from, as ...
- вложенность пакетов может быть любой (пакет в пакете ...)

Python

# Модули os, sys

# План

- Модуль os
- Модуль sys
- sys.path
- Практическое применение

# Модуль os

- Функции для работы с операционной системой
- Не зависит от конкретной ОС

# Функции и переменные os

- name - имя операционной системы
- chdir - смена текущей директории
- getcwd() - текущая рабочая директория
- mkdir() - создание директории (папки)
- os.path - модуль для работы с путями
- и многие другие

# Модуль sys

- Взаимодействие с интерпретатором Python

# Функции и переменные sys

- executable - путь к интерпретатору Python
- exit() - выход из Python
- platform - информация об ОС
- path - список путей поиска модулей
- argv - список аргументов командной строки
- и многие другие

# sys.path

- очень важная переменная
- она хранит пути? по которым Python ищет модули
- она имеет изменяемый тип данных list
- таким образом мы можем изменять эту переменную

# Пример для тренировки

- В папке с модулем создать 5 подпапок, названия которых состоят из платформы, на которой запущен интерпретатор и порядкового номера, начиная с 1: win32_1, win_32_2, … Платформа может быть другой.

Python

# Запуск скрипта с параметрами

# План

- sys.argv
- передача параметров в скрипт
- практическое применение

# sys.argv

- список аргументов командной строки при запуске скрипта Python
- sys.argv[0] - путь до скрипта
- остальные параметры передаются при вызове скрипта через пробел
- python my_script.py par1 par2 par3 ...

# Практический пример

- В зависимости от параметра вызывать различные функции скрипта
- Параметр ping -> функция выводит pong
- 2 параметра name <Имя> -> функция приветствия пользователя
- параметр list: показать содержимое текущей директории