Модули

№ урока:	7	**Курс:**	Python Essential
Средства обучения:	Python 3; интегрированная среда разработки (PyCharm или Microsoft Visual Studio + Python Tools for Visual Studio)		

Обзор, цель и назначение урока

После завершения урока обучающиеся будут иметь представление о модулях и пакетах и системе импортирования модулей в Python.

Изучив материал данного занятия, учащийся сможет:

- Создавать свои модули и объединять их в пакеты
- Импортировать модули, отдельные имена из модулей
- Пользоваться некоторыми стандартными модулями

Содержание урока

1. Понятие модулей
2. Импортирование модулей
3. Запуск модулей как скриптов
4. Пути поиска модулей
5. Файлы скомпилированного байт-кода модулей
6. Пакеты
7. Относительные импорты
8. Некоторые стандартные модули

Резюме

- *Модуль* — функционально законченный фрагмент программы, оформленный в виде отдельного файла с исходным кодом или поименованной непрерывной её части. Модули позволяют разбивать сложные задачи на более мелкие в соответствии с принципом модульности.
- Обычно проектируются таким образом, чтобы предоставлять программистам удобную для многократного использования функциональность (интерфейс) в виде набора функций, классов, констант.
- Модули могут объединяться в пакеты и, далее, в библиотеки.
- Удобство использования модульной архитектуры заключается в возможности обновления (замены) модуля, без необходимости изменения остальной системы
- *Модульное программирование* — это организация программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определенным правилам. Использование модульного программирования позволяет упростить тестирование программы и обнаружение ошибок. Аппаратно-зависимые подзадачи могут быть строго отделены от других подзадач, что улучшает мобильность создаваемых программ.
- Первым основные свойства программного модуля более-менее четко сформулировал Д. Парнас (David Parnas) в 1972 году: «Для написания одного модуля должно быть достаточно минимальных знаний о тексте другого». Таким образом, в соответствии с определением, модулем могла быть любая отдельная процедура (функция) как самого нижнего уровня иерархии (уровня реализации), так и самого верхнего уровня, на котором происходят только вызовы других процедур-модулей.
- Таким образом, Парнас первым выдвинул концепцию скрытия информации (англ. information hiding) в программировании. Однако существовавшие в языках 70-х годов только такие синтаксические конструкции, как процедура и функция, не могли обеспечить надежного скрытия информации, из-за повсеместного применения глобальных переменных.

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com

- Решить эту проблему можно было только разработав новую синтаксическую конструкцию, которая не подвержена влиянию глобальных переменных. Такая конструкция была создана и названа модулем. Изначально предполагалось, что при реализации сложных программных комплексов модуль должен использоваться наравне с процедурами и функциями как конструкция, объединяющая и надежно скрывающая детали реализации определенной подзадачи.
- Таким образом, количество модулей в комплексе должно определяться декомпозицией поставленной задачи на независимые подзадачи. В предельном случае модуль может использоваться даже для заключения в него всего лишь одной процедуры, если необходимо, чтобы выполняемое ею локальное действие было гарантировано независимым от влияния других частей программы при любых изменениях.
- Файл, который содержит исходный код на языке Python, является модулем.
- Название модуля доступно в его глобальной переменной __*name*__. Если модуль не импортирован, а запущен как скрипт, то __name__ устанавливается в значение “__main__”.
- Импортировать модули можно при помощи оператора *import*.
- Модули загружаются только один раз. Все последующие попытки их импортировать лишь возвращают ссылки на уже импортированные модули.
- При импортировании модуля выполняются все его операторы, находящиеся на уровне модуля.
- При импортировании модуля создаётся имя, соответствующее названию модуля, которое указывает на объект загруженного модуля. Указать другое имя можно при помощи ключевого слова *as*.
- Можно импортировать определённые имена из модуля, после чего они станут доступны как глобальные имена текущего модуля. Это делается при помощи оператора *from название_модуля import имена*. Если указано несколько имён, они разделяются запятыми. После каждого имени также можно указывать ключевое слово *as* и альтернативное имя.
- Если вместо списка имён указать символ *, то импортируются все имена, кроме тех, которые начинаются с символа подчёркивания.
- Для запуска модуля как скрипта его имя передаётся как параметр командной строки при вызове интерпретатора. После этого можно указывать другие параметры, которые передаются запускаемому скрипту и доступны как список argv, который находится во встроенном модуле sys.
- При импортировании модулей интерпретатор Python ищет их в директориях и архивах, список которых доступен как для чтения, так и для модификации в виде переменной path встроенного модуля sys.
- По умолчанию sys.path состоит из директории с запускаемым скриптом, содержимого переменной окружения PYTHONPATH и стандартного расположения модулей, специфичного для конкретной платформы и интерпретатора.
- Для ускорения загрузки модулей Python кеширует байт-код и производит компиляцию модуля только в том случае, если исходный код был изменён. Python 3 сохраняет файлы байт-кода .pyc в каталоге __pycache__, Python 2 – рядом с файлами исходного кода.
- Встроенная функция dir возвращает отсортированный список атрибутов модуля или любого другого объекта. Если не передать ей никаких параметров, то атрибуты текущего модуля.
- Встроенные классы, функции и переменные находятся в модуле builtins.
- Модули могут объединяться в *пакеты*. Пакеты служат как пространства имён для модулей и способ их структурирования.
- Любой пакет является модулем, но не каждый модуль является пакетом.
- Как правило, модули представляются в виде файлов, а пакеты – каталогов в файловой системе (но не всегда).
- Для того, чтобы каталог был пакетом, в нём должен находиться файл __init__.py. Он автоматически выполняется при импортировании соответствующего модуля и может содержать определённые действия для инициализации или быть пустым.
- При использовании оператора from package import item, item может быть пакетом, модулем или любым именем, описанным в пакете. При использовании оператора import package.item, item должен быть модулем или пакетом.

CyberBionic systematics

CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com

- Для того, чтобы можно было импортировать все имена пакета (from package.subpackage import *), пакет должен описывать список __all__, который содержит имена подпакетов и модулей.
- Существуют также относительные импорты: точка указывает на текущий пакет, две точки – на родительский. Эти же символы могут быть использованы сразу перед именем пакета или модуля и влиять на то, где интерпретатор будет его искать.
- Python имеет крайне богатую стандартную библиотеку. С её возможностями можно ознакомиться в документации.

Закрепление материала

- Что такое модуль?
- В чём смысл модульного программирования?
- Что такое пакет?
- Как импортировать модуль в Python?
- Как импортировать определённое имя из модуля?
- Как импортировать модуль из пакета?
- Как создать пакет?
- В чём разница между операторами import и from … import?

Дополнительное задание

Задание
Создайте модуль для получения простых чисел. Импортируйте его из другого модуля. Импортируйте отдельные его имена.

Самостоятельная деятельность учащегося

Задание 1
Перепишите домашнее задание предыдущего урока (сервис для сокращения ссылок) таким образом, чтобы у него была основная часть, которая отвечала бы за логику работы и предоставляла обобщённый интерфейс, и модуль представления, который отвечал бы за взаимодействие с пользователем. При замене последнего на другой, взаимодействующий с пользователем иным способом, программа должна продолжать корректно работать.

Задание 2
Повторите информацию о рассмотренных на уроке стандартных модулях. Ознакомьтесь также с модулями calendar, heapq, bisect, array, enum.

Рекомендуемые ресурсы

Документация Python
https://docs.python.org/3/tutorial/modules.html
https://docs.python.org/3/reference/import.html
https://docs.python.org/3/library/index.html
https://docs.python.org/3/library/datetime.html
https://docs.python.org/3/library/calendar.html
https://docs.python.org/3/library/heapq.html
https://docs.python.org/3/library/bisect.html
https://docs.python.org/3/library/array.html
https://docs.python.org/3/library/copy.html
https://docs.python.org/3/library/decimal.html
https://docs.python.org/3/library/fractions.html
https://docs.python.org/3/library/random.html

Статьи в Википедии о ключевых понятиях, рассмотренных на этом уроке
https://ru.wikipedia.org/wiki/Модуль_(программирование)

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site:www.edu.cbsystematics.com