АННОТАЦИЯ К РАБОЧЕЙ ПРОГРАММЕ ДИСЦИПЛИНЫ
«Программирование кроссплатформенных систем»

по основной профессиональной образовательной программе по направлению подготовки 12.03.01 «Приборостроение» (уровень бакалавриата)

Направленность (профиль): Искусственный интеллект в приборостроении
Общий объем дисциплины – 3 з.е. (108 часов)
Форма промежуточной аттестации – Экзамен.
В результате освоения дисциплины у обучающихся должны быть сформированы компетенции с соответствующими индикаторами их достижения:

- ПК-12.1: Разрабатывает программы и их блоки для построения интеллектуальных систем и приборов;
- ПК-12.2: Проводит отладку и настройку программ и программного обеспечения для построения интеллектуальных систем и приборов;

Содержание дисциплины:
Дисциплина «Программирование кроссплатформенных систем» включает в себя следующие разделы:
Форма обучения очная. Семестр 6.
1. Базовые концепции кросс-платформенного программирования. Понятие кроссплатформенного программного обеспечения. Проблема переносимости приложений с одной ОС на другую. Использование библиотек для создания графического интерфейса пользователя (ГИП или в англоязычной нотации GUI graphical user interface) и системных библиотек конкретной операционной системы (платформозависимых библиотек). Особенности кроссплатформенности java. Использование кроссплатформенных библиотек в стандартном языке (C, C++, Perl, Python, Ruby и др.) Веб-программирование как особый подход в кросс платформенном программировании. О кроссплатформенных IDE (Integrated development environment)..
2. Особенности программирования для различных операционных систем. Кроссплатформенные среды исполнения. Кроссплатформенные пользовательские интерфейсы. Особенности использования кросплатформенных библиотек на примере PyQt, PySide. Популярные фреймворки, которые помогут при разработке программ для компьютеров под управлением Windows/MacOS/Linux..
3. Подготовка к программированию на Python с использованием мобильных устройств. Установка и работа в Pydroid 3 на базе Android. Два основных подхода к работе с интерпретатором Python: непосредственная интерпретация строк кода, вводимых с клавиатуры в интерактивном режиме и выполнение файлов с исходным кодом в пакетном режиме. Вход в интерактивный режим работы. Пакетный режим работы. Python, как язык с неявной сильной динамической типизацией. Отличия динамической и статической типизации, сильной и слабой типизации. Разделение типов данных на встроенные в интерпретатор (built-in) и не встроенные, которые можно использовать при импортировании соответствующих модулей. Основные встроенные типы данных. Объявление и инициализация переменных. Объект, как абстракция для представления данных. Идентификатор как некоторое целочисленное значение, посредством которого уникально адресуется объект. Изменяемые и неизменяемые типы данных..
4. Арифметические операции. Работа со строками в Python. Три встроенных числовых типа данных. Арифметические операции с целыми и вещественными числами. Работа с комплексными числами. Доступные битовые операции в Python. Представление чисел в других системах счисления. Библиотека math из стандартной поставки Python. Литералы строк. Строки в

апострофах и в кавычках. Экранированные
последовательности для вставки служебных символов. Использование сырых
строк для подавления экранирования. Строки в тройных апострофах и
кавычках. Базовые операции для строк: конкатенация, дублирование строки,
определение длины строки, доступ по индексу, извлечение среза.
Форматирование строк с помощью метода format..

5. Условные операторы и циклы. Работа со списками. Конструкция условного оператора ветвления if. Использование
альтернативного варианта выполнения программы с помощью оператора if -
else. Реализация выбора из нескольких альтернатив можно с помощью
конструкции if - elif - else. Оператор цикла while для выполнения указанного
набора инструкций. Операторы break и continue для работы с циклами.
Оператор for для выполнения указанного набора инструкций заданное
количество раз. Список (list) как структура данных для хранения объектов различных
типов. Хранение списков в памяти. Создание, изменение, удаление списков и
работа с его элементами. Создание копий списков и простое присвоение
списков. Добавление и удаление элементов из списка. Методы списков.
"Списковое включение" как часть синтаксиса языка для предоставления
простого способа построения списков. List Comprehensions как обработчик
списков. Слайсы (срезы) - составляющая Python, которая позволяет быстро и
лаконично решать задачи выборки элементов из списка. Задание слайса
тройкой чисел, разделенных запятой: start:stop:step ..

6. Кортежи. Словари. Функции в Python. Кортеж (tuple) как неизменяемая структура данных. Причины, по
которым стоит использовать кортежи вместо списков. Создание, удаление
кортежей и работа с его элементами. Преобразование кортежа в список и
обратно. Структура данных, предназначенная для хранения произвольных
объектов с доступом по ключу. Хранение данных в формате ключ - значение.
Создание, изменение, удаление словарей и работа с его элементами. Проверка
наличия ключа в словаре. Методы словарей. Функция как именованный фрагмент программного кода, к которому
можно обратиться из другого места программы. Использование ключевого
слова def для создания функции. Возврат значения функцией. Использование
функций для обработки данных. Безымянная функция с произвольным числом
аргументов..

7. Работа с исключениями. Ввод-вывод данных. Работа с файлами. Понятие исключения в языке программирования. Как исключения дают
возможность дальнейшей работы в рамках основного алгоритма. Синхронные
и асинхронные исключения. Структурная и неструктурная обработка
исключений. Синтаксические ошибки. Иерархия исключений в Python.
Обработка исключения внутри синтаксической конструкции try...except.
Выполнение определенного программного кода при выходе из блока try/except
с помощью оператора finally . Генерация исключений в Python.
Пользовательские исключения.
Вывод данных в консоль. Функция для считывания вводимых с
клавиатуры данных. Преобразование строки в список с помощью метода split()
. Считывание списка чисел с одновременным приведением их к типу int .
Открытие и закрытие файла. Чтение данных из файла. Запись данных в файл.
Дополнительные методы для работы с файлами..

8. Классы и объекты. Итераторы и генераторы. Три основных "столпа" ООП - инкапсуляция, наследование и
полиморфизм. Создание класса в Python с помощью инструкции class.
Объявление класса, имя класса и тело класса. Создание объекта класса.
Статические и динамические атрибуты класса. Методы класса: статические,

классовыми (среднее между статическими и обычными) и уровня класса. Конструктор класса и инициализация экземпляра класса. Ключевое слово se lf. Уровни доступа атрибута и метода в Python. Свойство как метод класса, работа с которым подобна работе с атрибутом. Наследование. Родительский класс. Переопределение методов базового класса в классе наследнике на базе полиморфизма. Инструменты, которые, как правило, используются для поточной обработки данных. Использование итератора для упрощения навигации по элементам объекта. Создание собственных итераторов. Упрощение работы по конструированию итераторов с помощью генераторов..

Разработал:
заведующий кафедрой
кафедры ИТ А.Г. Зрюмова

Проверил:
Декан ФИТ А.С. Авдеев