Universidad de Navarra

Python basics

Guía docente 2026-27

PRESENTACIÓN

Utilizar el ordenador para el análisis de datos es un valioso complemento a la formación científica. El lenguaje de programación ***Python*** está especialmente indicado para esa tarea. En esta asignatura se estudiará la *sintaxis* de este lenguaje de programación (es decir, los comandos y las expresiones, que vienen a ser el vocabulario y la gramática). Además, se expondrá cómo plasmar un algoritmo en un esquema básico (*pseudocódigo*), con especial atención a los mecanismos de tomas de decisiones (condiciones), para después implementarlo en un programa. El curso es eminentemente práctico.

- Titulación**: Diploma en Ciencia de Datos**
- ECTS**: 3**
- Curso**: 1º,** semestre**: 2º**
- Profesor**: Angel Garcimartín**
- Idioma**: Español**
- Aula, Horario: *Se indicará en su momento*

PROGRAMA

1. **Introducción e instalación**
    1. Por qué Python: visión general y objetivos del curso
    2. Instalación de Python (conda) y de Jupyter Notebook
    3. Interfaz de usuario; edición en Markdown
    4. Bibliografía, ayuda y recursos
2. **Sintaxis de Python, operadores y tipos de variables**
    1. Reglas sintácticas, variables, asignación
    2. Operadores aritméticos
    3. Condiciones y operadores lógicos
    4. Entrada y salida básica
3. **Colecciones: listas, tuplas, diccionarios y sets**
    1. Definiciones, índices y secciones
    2. Métodos para listas
    3. Métodos para diccionarios
4. **Cadenas de caracteres**
    1. Formateo del texto
    2. Métodos para cadenas de caracteres
5. **Funciones**
    1. Definición y llamada; argumentos
    2. Módulos
    3. Funciones lambda
6. **Control de flujo (I) : ramificación**
    1. Pseudocódigo
    2. Condiciones: lógica de Boole
    3. Expresiones básicas: if, elif, else
7. **Control de flujo (II) : bucles**
    1. Pseudocódigo
    2. Iteraciones
    3. Comandos in, range
    4. Bucles while y for
    5. *List comprehension*
8. **Conclusiones y perspectivas**

**Nota*.- Parte del temario puede ser abreviado dependiendo de la marcha del curso.

ACTIVIDADES FORMATIVAS

Universidad de Navarra

El curso será eminentemente práctico (*learn by doing*); **hay que acudir a clase con ordenador portátil**.

La asistencia a clase no es obligatoria en sí misma, pero dado que se calificarán las actividades ordinarias (evaluación continua) la falta de asistencia puede penalizar estas calificaciones.

- Sesiones teóricas: 12 horas
- Sesiones prácticas (resolución de problemas, casos prácticos, etc.): 18 horas
- Tutorías: *ad lib*.

EVALUACIÓN

CONVOCATORIA ORDINARIA

1. Evaluación continua: notas de clase, exposición pública de ejercicios resueltos y casos prácticos: 20%
2. Pruebas cortas realizadas durante el curso (tests, ejercicios prácticos, etc.): 20%
3. Examen final (teoría y resolución de ejercicios prácticos): 60%

CONVOCATORIA EXTRAORDINARIA

- Las notas de la evaluación continua y de las pruebas cortas (ítems 1 y 2) se guardan para la convocatoria extraordinaria.
- Examen final (teoría y resolución de ejercicios prácticos): 60%

Cualquier estudiante puede ser llamado a examen oral para explicar o clarificar las respuestas que haya dado en las pruebas escritas o en los ejercicios y casos prácticos presentados.

HORARIOS DE ATENCIÓN

Dr. Angel Garcimartín (angel@unav.es)

- Despacho O-140, Departamento de Física, Edificio Los Castaños
- Horario de tutoria: *a convenir con los alumnos (se fijará al comenzar el curso)*

BIBLIOGRAFÍA Y RECURSOS

La mayoría de la ayuda y la documentación que se utilizará se encuentra disponible en Internet y se presentará durante el curso.

Algunos manuales que pueden servir de ayuda son los siguientes:

- Raúl González Duque, "**Python para todos**", libro gratuito disponible en internet [enlace]
- Jake VanderPlas, "**A whirlwind tour of Python**", O'Reilly (con una versión pdf gratuita en este enlace)
- A. Garcimartín, "**Introducción a Python para cálculo científico**", apuntes disponibles en internet [enlace]

RESULTADOS DE APRENDIZAJE (Competencias)

- Conocer la sintaxis básica de Python
- Saber trabajar en una interfaz de programación y buscar ayuda de forma autónoma
- Capacidad para plasmar un algoritmo en un esquema (pseudocódigo) y traducirlo a un programa
- Entender y utilizar los mecanismos de toma de decisiones (control del flujo)