PROGRAMACIÓN CON PYTHON

CURSOS

PH1: Programación con PYTHON nivel I

PRESENCIAL: 80 horas
ONLINE: 30 horas Clases en Directo + 50 horas de Proyectos Tutorizados

PH2: Programación con PYTHON nivel II

PRESENCIAL: 80 horas
ONLINE: 30 horas Clases en Directo + 50 horas de Proyectos Tutorizados

MÁSTERS

MPH: MASTER Programación con PYTHON (PH1 + PH2)

PRESENCIAL: 160 horas
ONLINE: 60 horas Clases en Directo + 100 horas de Proyectos Tutorizados

Deica IT
Desarrollo y Formación

PROGRAMACIÓN CON PYTHON

OBJETIVOS

La programación con Python permite un desarrollo de código y prototipado muy rápido y cuenta con un gran número de librerías en constante crecimiento debido a su gran utilización. Es usado en ámbitos muy diferentes como pueden ser administración de sistemas, diseño web, software testing, aplicaciones científicas y matemáticas, creación de aplicaciones, diseño gráfico, juegos, arquitectura o seguridad informática. Además puede ser usado en cualquier sistema operativo, lo que hace que sea un lenguaje muy demandado por las empresas.

A QUIÉN VA DIRIGIDO

Los cursos están dirigidos a todos aquellos alumnos y profesionales del sector que necesiten aprender a desarrollar aplicaciones de escritorio y Web y quieran adquirir los conocimientos necesarios que actualmente demandan las empresas para trabajar en el desarrollo y programación de todo tipo de aplicaciones bajo plataforma PYTHON.

MODALIDADES

Presencial u Online

REQUISITOS

El plan de estudios asume que los estudiantes tengan conocimientos previos de informática a nivel de usuario. Se espera de ellos una buena capacidad de lectura y expresión escrita, así como un deseo de aprender el programa de estudios:

- Imprescindibles:
  + Conocimientos a nivel usuario del Sistema operativo Windows
  + Disponer de horas adicionales (entre 5 y 10) a la semana para realizar ejercicios

Deica IT
Desarrollo y Formación

Programación

PH1: Programación con PYTHON nivel I

DURACIÓN

80 HORAS

PROGRAMA

Introducción

- ¿Por qué aprender Python?: ventajas y desventajas
- Características del lenguaje
- Casos de uso
- Instalación y configuración de Python 3
- Instalación y presentación del IDE
- Introduccion al intérprete de Python
- Cómo lanzar programas de Python

Tipos y Operaciones

- Tipos numericos: formatos, operaciones básicas, comparaciones
- Tipado dinámico en Python
- Cadenas de caracteres (Strings): conversiones, métodos, formateo
- Listas, tuplas, diccionarios: crear, modificar y recorrer estas estructuras
- Ficheros: manejo y formatos

Programación

- Otros tipos de Python
- Ejercicios

Sintaxis y sentencias

- Revisión de la jerarquía de Python
- Ejemplo sencillo: bucles interactivos
- Asignación, expresiones e imprimir por pantalla
- Sentencia condicional if
- Bucles for y while
- Técnicas de programación de bucles
- Iteraciones e Introduccion a las listas por comprensión
- Documentación de código
- Ejercicios

Funciones

- Definición, llamadas y polimorfismo de funciones en Python
- Ambito de los elementos de Python
- Argumentos de Funciones
- Temas avanzados de funciones

Módulos y paquetes

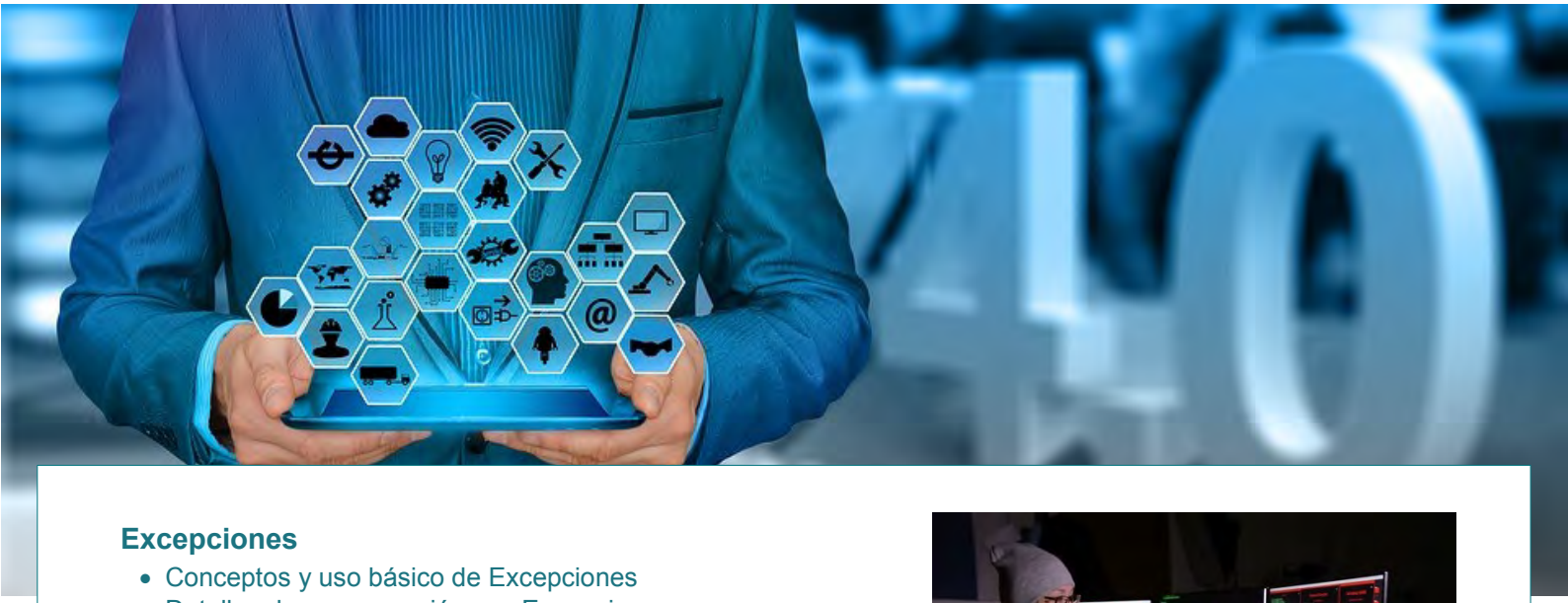
- Introduccion a los módulos
- Importación y uso de módulos
- Espacio de nombres de los módulos
- Paquetes de módulos

Programación Orientada a Objetos

- Introduccion a las clases y Objetos
- Programación básica de clases
- Instancias, métodos y atributos
- Herencia
- Espacio de nombres de clases
- Sobrecarga de operadores

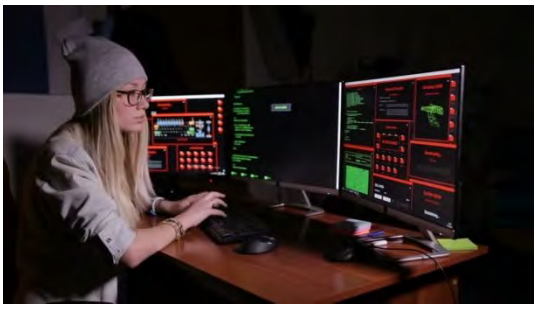
Deica IT
Desarrollo y Formación

Programación



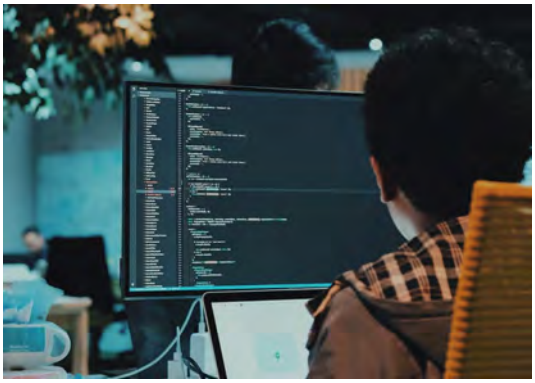
Excepciones

- Conceptos y uso básico de Excepciones
- Detalles de programación con Excepciones



Estructuras de datos y algoritmos

- Desempaquetar una secuencia en variables separadas
- Desempaquetar elementos de iterables de cualquier longitud
- Guardar los últimos N elementos
- Encontrar los N mayores o menores elementos
- Implementar una cola con prioridad
- Mapear claves a múltiples valores en un diccionario
- Diccionarios ordenados
- Cálculos con diccionarios
- Encontrar partes comunes en diccionarios
- Eliminar duplicados en una secuencia manteniendo el orden
- Renombrar slices de secuencias
- Obtener los elementos que más se repiten en una secuencia
- Ordenar una lista de diccionarios por una clave común
- Ordenar objetos que no tienen tienen comparación soportada nativamente
- Agrupar registros basándose en un campo
- Filtrar elementos de una secuencia
- Extraer un subconjunto de un diccionario
- Mapear nombres a elementos de secuencias
- Transformar y reducir datos al mismo tiempo
- Combinar varios diccionarios en uno



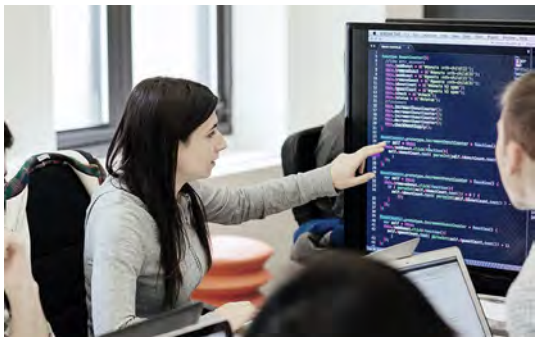
Cadenas y texto

- Separar cadenas usando cualquiera de múltiples delimitadores
- Encontrar texto al principio o final de una cadena
- Encontrar cadenas usando patrones Wildcard tipo

Programación

Shell

- Buscar patrones de texto
- Búsqueda y sustitución de texto
- Búsqueda y sustitución de texto ignorando las mayúsculas
- Especificar expresión regular para la ocurrencia menor posible
- Escribir expresiones regulares para múltiples patrones
- Normalizar texto unicode a una representación estándar
- Trabajar con caracteres unicode en expresiones regulares
- Eliminar caracteres no deseados de cadenas
- Saneamiento de texto
- Alinear cadenas de texto
- Combinar y concatenar cadenas
- Interpolar variables dentro de cadenas
- Re-formatear texto a un número fijo de columnas
- Manejar entidades HTML y XML en un texto
- Convertir texto a símbolos
- Escribir un parseador recursivo
- Realizar operaciones de texto en Cadenas de Bytes



Números, Fechas y Horas

- Redondear valores numéricos
- Realizar operaciones decimales precisas
- Formatear numero para mostrar
- Trabajar con enteros Binarios, Octales y Hexadecimales
- Empaquetar y desempaquetar Enteros Largos a Bytes
- Operaciones matemáticas con complejos
- Trabajar con Infinito y NaNs
- Cálculos con fracciones
- Cálculos con arrays numéricos grandes
- Cálculos con matrices y álgebra lineal
- Números aleatorios
- Conversiones de tiempo

Deica IT
Desarrollo y Formación

Programación

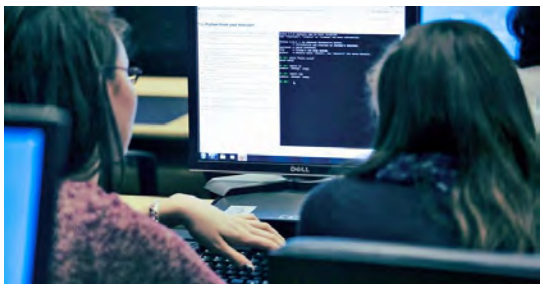
- Saber la fecha del Ultimo Viernes
- Saber el rango de fechas de un mes
- Convertir cadenas a Datetimes
- Manipular fechas y horas de reuniones teniendo en cuenta la zona horaria

Iteradores y generadores

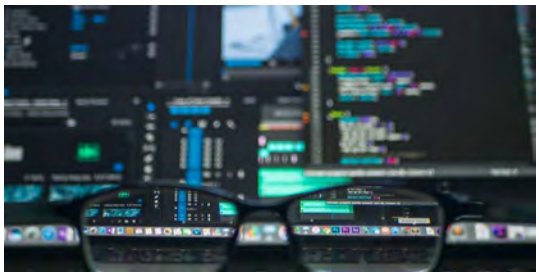
- Consumir manualmente un iterador que no queremos recorrer con un bucle
- Delegar iteración, crear un objeto que contiene otro iterable en su interior y hacer que el nuevo objeto sea iterable
- Crear nuevos patrones de iteración con generadores
- Implementar el protocolo del iterador
- Iteración en orden inverso
- Definir funciones generadoras con estados adicionales
- Tomar porciones(slices) de generadores
- Saltarse la primera parte de un iterable
- Iterar sobre todas las posibles combinaciones o permutaciones de una colección de objetos
- Iterar sobre paras indice-valor de una secuencia
- Iterar sobre múltiples secuencias simultáneamente
- Iterar sobre elementos en diferentes contenedores
- Crear pipelines de procesado de datos
- Aplanar secuencias encadenadas(de varias dimensiones)
- Iterar de manera ordenada sobre una secuencia creada a partir de varias secuencias ordenadas
- Sustituir bucles while infinitos por un iterador

Ficheros y Entrada/Salida

- Leer y escribir texto
- Imprimir a un fichero
- Imprimir con diferentes separadores y finales de linea
- Leer y escribir datos binarios
- Escribir a un fichero que todavía no existe
- Realizar operaciones de entrada y salida con cadenas en lugar de con ficheros



Programación



- Leer y escribir ficheros comprimidos
- Iterar sobre partes fijas de ficheros en lugar de sobre lineas
- Leer datos binarios de un buffer sin hacer copias intermedias
- Mapear ficheros binarios en memoria como arrays de bytes
- Manipular nombres y rutas de ficheros
- Comprobar la existencia de un fichero
- Obtener el listado de todos los ficheros de un directorio
- Realizar operaciones de E/S con ficheros que no han sido codificados de acuerdo al sistema de codificación de nombres de fichero por defecto
- Imprimir nombres de ficheros corruptos
- Añadir o cambiar la codificación de un fichero que ya esta abierto
- Escribir bytes a un fichero de texto
- Convertir un descriptor de fichero existente a un objeto fichero de alto nivel de python
- Crear ficheros y directorios temporales
- Comunicación con puerto serie
- Serializar objetos

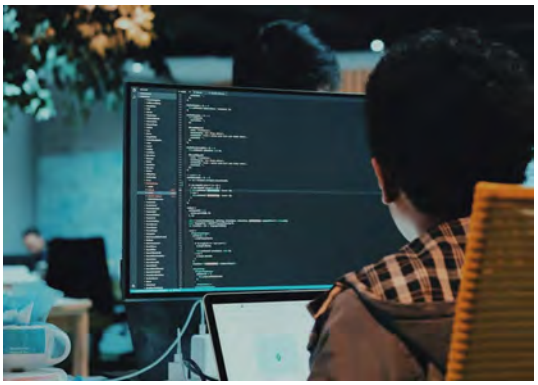
Deica IT
Desarrollo y Formación

Programación

PH2: Programación con PYTHON nivel II

DURACIÓN

80 HORAS



PROGRAMA

Codificación y procesado de datos

- Leer y escribir datos CSV
- Leer y escribir datos JSON
- Parsear datos XML simples
- Parsear ficheros XML muy grandes de forma incremental
- Convertir un diccionario en XML
- Parsear, modificar y reescribir XML
- Parsear documentos XML con Namespaces
- Interactuar con bases de datos relacionales
- Decodificar y codificar números hexadecimales
- Decodificar y codificar números en Base64
- Leer y escribir arrays de estructuras de datos binarios
- Agregar datos y realizar estadísticas

Funciones avanzadas

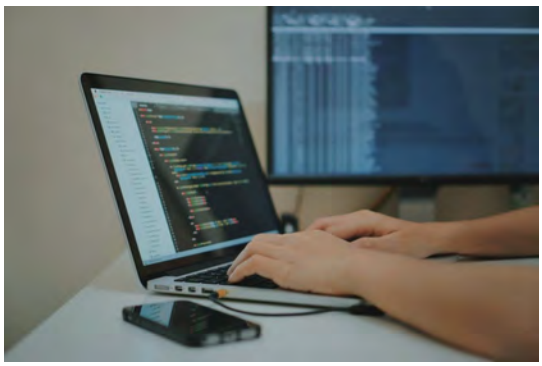
- Escribir funciones que acepten cualquier numero de argumentos
- Escribir funciones que solo acepten argumentos tipo

Programación

clave-valor
- Añadir metadados de información a los argumentos de una función
- Devolver múltiples valores en una función
- Definir funciones con argumentos por defecto
- Definir funciones anónimas
- Capturar variables en funciones anónimas
- Hacer que una función con N argumentos se comporte como si tuviera menos
- Sustituir clases con un sólo método por funciones
- Pasar estado adicional a las funciones callback
- Hacer que una callback se comporte como una serie de pasos procedimentales
- Acceder a variables definidas dentro de una clausura

Clases y objetos: conceptos y técnicas avanzadas

- Cambiar la representación que obtenemos al imprimir una instancia
- Personalizar el formateado de cadenas
- Hacer que un objeto soporte el protocolo de manejo contexto(with statement)
- Ahorrar memoria al crear un numero grande de instancias
- Encapsular nombres en una clase
- Invocar un método en la clase padre
- Extender una propiedad en una subclase
- Crear un nuevo tipo de atributo de clase o instancia
- Crear propiedades que solo se evalúen al acceder a ellas(Lazily Computed)
- Simplificar la inicialización de estructuras de datos
- Definir el interfaz de una clase base abstracta
- Implementar un modelo de datos
- Implementar contenedores comunes
- Delegar el acceso a atributos
- Definir mas de un constructor en una clase
- Crear una instancia sin invocar el método init
- Extender clases con métodos de otras clases
- Implementar máquinas de estados



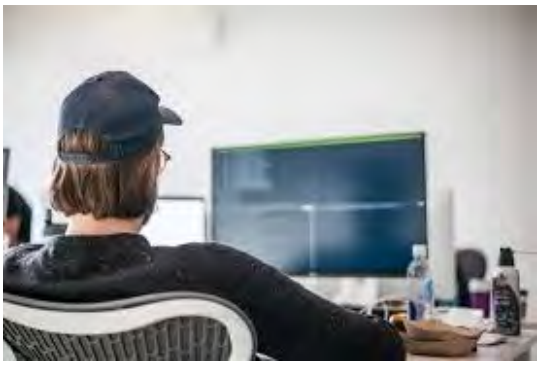
- Llamar un método en un objeto dado su nombre como cadena de texto
- Implementar el patrón visitor
- Implementar el patrón visitor sin recursividad
- Manejo de memoria en estructuras de datos cíclicas
- Hacer operaciones de comparación que soporten clases
- Crear instancias guardadas en caché

Metaprogramación

- Poniendo envolturas(wrappers) a funciones
- Guardar metadatos de una función al escribir decoradores
- Eliminando un decorador
- Definir un decorador que tome argumentos
- Definir un decorador con atributos ajustables por el usuario
- Definir un decorador con un argumento opcional
- Forzar el chequeo de tipo en una función usando decoradores
- Definir un decorador como parte de una clase
- Definir decoradores como clases
- Aplicar un decorador a un método estático y a una clase
- Escribir decoradores que añadan argumentos a funciones con envolturas
- Usar decoradores para modificar definiciones de clases
- Usar una metaclase para controlar la creación de instancias
- Capturar el orden de definición de atributo de clase
- Definir una metaclase que tome argumentos adicionales
- Forzar la firma de argumentos en *args y **kwargs
- Forzar convenciones de código en clases
- Definir clases programáticamente
- Inicializar miembros de las clases en tiempo de definición
- Implementar envió múltiple con anotación de funcio-

Programación

nes
- Evitar métodos repetitivos
- Definir gestores de contexto de manera sencilla
- Ejecutar código con efectos laterales locales
- Parsear y analizar código fuente python
- Desensamblar código python



Módulos y paquetes: técnicas de programación

- Crear un paquete jerárquico de módulos
- Controlar todas las importaciones
- Importar submódulos de un paquete usando nombres relativos
- Partir módulos en varios ficheros
- Crear directorios separados de importación de código bajo un espacio de nombres comun
- Recargando módulos
- Hacer que un directorio o fichero comprimido sea ejecutable como si fuera un script principal
- Leyendo Datafiles dentro de un paquete
- Añadiendo directorios a sys.path
- Importar módulos usando un nombre dado como cadena de texto
- Importar módulos de una maquina remota usando ganchos de importación
- Modificar módulos al importarlos
- Instalar paquetes solo para uno mismo
- Crear un nuevo entorno de Python
- Distribuir paquetes

Concurrencia

- Arrancar y parar hilos
- Determinar si un hilo ha arrancado
- Comunicación entre hilos
- Proteger secciones criticas
- Protección de secciones criticas evitando interbloqueo
- Almacenar el estado de los hilos

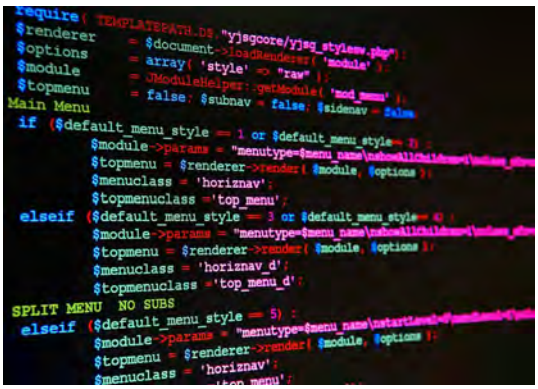
Deica IT
Desarrollo y Formación

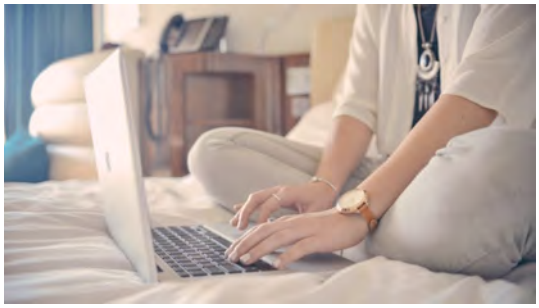
Programación

- Crear un fondo de hilos
- Ralizar programación en paralelo
- El Global interpreter Lock (GIL)
- Definir una tarea de actor
- Implementar mensagería de publicar/suscribir
- Usar generadores como una alternativa a los hilos
- Preguntar a colas de múltiples hilos
- Arrancar un demonio en Unix

Administración de sistemas: scripts con Python y Testeo/depuración de código

- Aceptar dados de entrada para un script mediante redirección, tuberias y ficheros de entrada
- Terminar un programa con un mensaje de error
- Parsear opciones por linea de comandos
- Pedir contraseña en tiempo de ejecución
- Averiguar el tamaño del terminal
- Ejecutar un comando externo y obtener su salida
- Copiar y mover ficheros y directorios
- Crear y desempaquetar ficheros
- Encontrar ficheros por nombre
- Leer ficheros de configuración
- Añadir logging a scripts y librerias
- Hacer un cronometro
- Poner limites al consumo de memoria y CPU
- Lanzar un navegador Web
- Testear la salida enviada a stdout
- Modificar objetos en tests unitarios
- Testear buscando condiciones excepcionales en tests unitarios
- Guardar la salida de tests a ficheros
- Evitar o anticiparse a fallos de tests
- Manejar múltiples excepciones
- Capturar todas las excepciones
- Lanzar una excepcion en respuesta a otra excepción
- Relanzar la ultima excepción
- Lanzar mensajes de aviso
- Depurar roturas básicas de programas





Programación

- Obtener consumo de recursos y tiempo de un programa
- Hacer que un programa corra mas rápido

Programación para Web y redes con Python. Análisis de Datos

- Interactuar con servicios HTTP
- Introducción al Web Scraping