

Podrá descargar algunos elementos de este libro en la página web
de Ediciones ENI: <http://www.ediciones-eni.com>.
Escriba la referencia ENI del libro **RITHS-43PYT** en la zona de búsqueda
y valide. Haga clic en el título y después en el botón de descarga.

Prólogo

1. Introducción	23
2. Contenido del libro	23
3. Progresión del libro	24
4. Destinado a profesores y alumnos	25
5. Destinado a investigadores y doctores	26
6. Destinado a aquellos que vienen de otro lenguaje	27

Parte 1: Las fortalezas de Python

Capítulo 1-1

Claves teóricas

1. Breve historia de los lenguajes informáticos	29
1.1 Informática teórica	29
1.2 Cronología de la informática	30
1.2.1 Evolución de las problemáticas vinculadas a la informática . . .	30
1.2.2 Cronología de los lenguajes informáticos	31
2. Tipología de los lenguajes de programación	35
2.1 Paradigmas	35
2.1.1 Definición	35
2.1.2 Paradigma imperativo y derivados	36
2.1.3 Paradigma orientado a objetos y derivados	37
2.1.4 Programación orientada a aspectos	37
2.1.5 Paradigma funcional	38
2.1.6 Paradigma lógico	38
2.1.7 Programación concurrente	38
2.1.8 Síntesis	39
2.2 Interoperabilidad	39
2.3 Niveles de programación	41
2.3.1 Máquina	41
2.3.2 Bajo nivel	41
2.3.3 Alto nivel	42

2 **Python 3**

Los fundamentos del lenguaje

2.4	Tipado	43
2.4.1	Débil vs. fuerte	43
2.4.2	Estático vs dinámico	43
2.5	Gramática	43
2.5.1	Lenguajes formales	43
2.5.2	Sintaxis	44
3.	Python y el resto del mundo	45
3.1	Posición estratégica del lenguaje Python	45
3.1.1	Segmentos de mercado	45
3.1.2	Nivel de complejidad	45
3.1.3	Fortalezas del lenguaje	45
3.1.4	Puntos débiles	46
3.2	Integración con otros lenguajes	46
3.2.1	Extensiones C	46
3.2.2	Integración de programas escritos en C	47
3.2.3	Integración de programas Python en C	47
3.2.4	Integración de programas escritos en Java	47
3.2.5	Integración de programas Python en Java	47
3.2.6	Otras integraciones	47

Capítulo 1-2 Presentación de Python

1.	Filosofía	49
1.1	Python en pocas líneas	49
1.1.1	¿De dónde proviene el nombre «Python»?	49
1.1.2	Presentación técnica	49
1.1.3	Presentación conceptual	50
1.2	Comparación con otros lenguajes	50
1.2.1	Shell	50
1.2.2	Perl	51
1.2.3	C, C++	51
1.2.4	Java	52
1.2.5	PHP	54
1.3	Grandes principios	55
1.3.1	El zen de Python	55
1.3.2	El desarrollador no es estúpido	55
1.3.3	Documentación	56
1.3.4	Python viene con todo incluido	56
1.3.5	Duck Typing	56

1.3.6	Noción de código pythónico	57
2.	Historia de Python.	57
2.1	La génesis.	57
2.2	Ampliación del perímetro funcional	58
2.3	Evolución de la licencia.	62
2.4	El futuro	62
3.	Gobierno.	63
3.1	Desarrollo	63
3.1.1	Ramas.	63
3.1.2	Comunidad	64
3.2	Modo de gobierno.	64
3.2.1	Creador del lenguaje	64
3.2.2	PEP	65
3.2.3	Toma de decisiones	65
3.2.4	Contribuir a Python	65
4.	¿Qué contiene Python?	66
4.1	Una gramática y una sintaxis	66
4.2	Varias implementaciones	66
4.3	Una librería estándar	66
4.4	Librerías de terceros	67
4.5	Frameworks.	67
5.	Fases de ejecución de un programa Python	67
5.1	Carga de la máquina virtual.	67
5.2	Compilación	67
5.3	Interpretación	68

Capítulo 1-3

Por qué escoger Python

1.	Cualidades del lenguaje.	69
1.1	Puerta de entrada	69
1.2	Cualidades intrínsecas	71
1.3	Cobertura funcional	71
1.4	Dominios de excelencia	72
1.5	Garantías.	73
2.	Difusión	74
2.1	Empresas	74
2.2	El mundo de la investigación	75

4 Python 3

Los fundamentos del lenguaje

2.3	El mundo de la educación	76
2.4	Comunidad	77
3.	Referencias	78
3.1	Pesos pesados en la industria informática	78
3.1.1	Google	78
3.1.2	Mozilla	79
3.1.3	Microsoft	79
3.1.4	Canonical	79
3.1.5	Cisco	80
3.2	Empresas de innovación	80
3.2.1	Servicios de almacenamiento en línea	80
3.2.2	Cloud computing	80
3.2.3	Plataforma colaborativa (Forge)	80
3.2.4	Redes sociales	80
3.3	Editores de contenidos	81
3.3.1	Disney Animation Studio	81
3.3.2	YouTube	81
3.3.3	Box ADSL	81
3.3.4	Spotify	81
3.4	Fabricantes de software	81
4.	Experiencia	82
4.1	Internet de las cosas	82
4.2	Sistemas y desarrollo web	83
4.3	Enseñanza	84
4.4	Informática embebida	84
4.5	Desarrollo web	85
4.6	ERP	85

Capítulo 1-4

Instalar el entorno de trabajo

1.	Introducción	87
2.	Instalar Python	87
2.1	Para Windows	87
2.2	Para Mac	90
2.3	Para GNU/Linux et BSD	90
2.4	Mediante compilación	91
2.5	Para un smartphone	92

3.	Instalar una librería externa	92
3.1	A partir de Python 3.4	92
3.2	Para una versión inferior a Python 3.4	94
3.3	Para Linux	94
4.	Crear un entorno virtual	95
4.1	¿Para qué sirve un entorno virtual?	95
4.2	Para Python 3.3 o versiones superiores	95
4.3	Para cualquier versión de Python	96
4.4	Para Linux	97
5.	Gestión de las dependencias	98
6.	Instalar Anaconda	99
6.1	Para Windows	99
6.2	Para Linux	100
6.3	Para Mac	100
6.4	Actualizar Anaconda	100
6.5	Instalar una librería externa	100
6.6	Entornos virtuales	101
7.	Docker	101
8.	La consola Python	102
8.1	Arrancar la consola Python	102
8.2	BPython	102
8.3	IPython	103
8.4	IPython Notebook	103
9.	Instalar un IDE	104
9.1	Lista de IDE	104
9.2	Presentación de PyCharm	104
9.3	Configuración de PyCharm	105
10.	VSCoDe	110

Parte 2: Guía de Python

Capítulo 2-1

Primeros pasos

1.	Antes de comenzar	111
1.1	Algunas nociones importantes	111
1.1.1	¿Cómo funciona un ordenador?	111

6 **Python 3**

Los fundamentos del lenguaje

1.1.2	¿Qué es un programa informático?	112
1.1.3	¿Qué es el código fuente?	112
1.2	Algunas convenciones utilizadas en este libro	112
1.2.1	Código Python	112
1.2.2	Terminal	113
1.2.3	Formato	113
1.3	¿Cuál es el mejor método para aprender?	114
2.	Primer programa	114
2.1	Hello world!	114
2.2	Asignación	116
2.3	Valor booleano	117
2.4	Tipo	118
2.5	Excepciones	119
2.6	Bloque condicional	121
2.7	Condiciones avanzadas	123
2.8	Bloque iterativo	123
3.	Primer juego: Adivine el número	125
3.1	Descripción del juego	125
3.2	Pistas	125
3.2.1	Gestión del azar	125
3.2.2	Etapas del desarrollo	126
3.3	Para ir más allá	126

Capítulo 2-2

Funciones y módulos

1.	Las funciones	127
1.1	¿Por qué utilizar funciones?	127
1.2	Introducción a las funciones	129
1.2.1	Cómo declarar una función	129
1.2.2	Gestión de un parámetro	130
1.2.3	Cómo hacer la función más genérica	132
1.2.4	Parámetros por defecto	134
1.3	Problemáticas de acoplamiento y duplicación de código	135
1.3.1	Nivel de sus funciones	135
1.3.2	Noción de complejidad	137
1.3.3	Buenas prácticas	139

2.	Los módulos	140
2.1	Introducción	140
2.1.1	¿Qué es un módulo?	140
2.1.2	¿Cómo crear un módulo en Python?	141
2.1.3	Organizar el código	141
2.2	Gestionar el código de los módulos	141
2.2.1	Ejecutar un módulo, importar un módulo	141
2.2.2	Gestionar un árbol de módulos	142
3.	Terminar el juego.	143
3.1	Crear niveles	143
3.2	Determinar un número máximo de intentos	144
3.3	Registrar las mejores puntuaciones	144
3.4	Inteligencia artificial.	144

Capítulo 2-3

Los principales tipos

1.	Cadenas de caracteres	145
1.1	Sintaxis	145
1.2	Formato de una cadena	146
1.3	Noción de tamaño de letra.	146
1.4	Noción de longitud	147
1.5	Pertenencia	148
1.6	Noción de ocurrencia	148
1.7	Reemplazo.	149
1.8	Noción de carácter	149
1.9	Tipología de los caracteres	150
1.10	Secuenciar una cadena de caracteres	151
2.	Listas.	152
2.1	Sintaxis	152
2.2	Índices	152
2.3	Valores.	153
2.4	Azar	154
2.5	Técnicas de iteración	155
2.6	Ordenación	158
3.	Diccionarios	159
3.1	Presentación de los diccionarios.	159
3.2	Recorrer un diccionario	160

3.3 Ejemplo	160
-------------------	-----

Capítulo 2-4

Las clases

1. Sintaxis	163
2. Noción de instancia en curso	164
3. Operadores	167
4. Herencia	168
4.1 Especialización	169
4.2 Programación por composición	170

Parte 3: Los fundamentos del lenguaje

Capítulo 3-1

Algoritmos básicos

1. Delimitadores	173
1.1 Instrucción	173
1.2 Una línea de código = una instrucción	173
1.3 Comentario	174
1.4 Una instrucción en varias líneas	174
1.5 Palabras clave	174
1.6 Palabras reservadas	175
1.7 Indentación	176
1.8 Símbolos	177
1.9 Operadores	180
1.9.1 Operador de expresión de asignación :=	183
1.10 Uso del carácter de subrayado	184
1.11 PEP-8	185
1.12 PEP-7	185
1.13 PEP-257	185
2. Instrucciones	186
2.1 Definiciones	186
2.1.1 Variable	186
2.1.2 Función	188
2.1.3 Funciones lambda	189
2.1.4 Clase	189
2.1.5 Instrucción vacía	190

2.1.6	Borrado	191
2.1.7	Devolver el resultado de la función	191
2.2	Instrucciones condicionales	192
2.2.1	Definición	192
2.2.2	Condición	193
2.2.3	Instrucción if	193
2.2.4	Instrucción elif	193
2.2.5	Instrucción else	194
2.3	Instrucción de correspondencia	195
2.4	Uso de una expresión de asignación	198
2.4.1	Instrucción switch	199
2.4.2	Interrupciones	199
2.4.3	Profundizando en las condiciones	199
2.4.4	Rendimiento	200
2.5	Iteraciones	201
2.5.1	Instrucción for	201
2.5.2	Instrucción while	202
2.5.3	¿Cuál es la diferencia entre for y while?	202
2.5.4	Instrucción break	203
2.5.5	Instrucción return	204
2.5.6	Instrucción continue	205
2.5.7	Instrucción else	205
2.5.8	Generadores	206
2.6	Construcciones funcionales	209
2.6.1	Construcción condicional	209
2.6.2	Generadores	209
2.6.3	Recorrido de listas	209
2.6.4	Recorrido de conjuntos	209
2.6.5	Recorrido de diccionarios	210
2.7	Recorrido y expresión de asignación	210
2.8	Gestión de excepciones	210
2.8.1	Breve presentación de las excepciones	210
2.8.2	Elevar una excepción	211
2.8.3	¿Por qué elevar una excepción?	211
2.8.4	Aserciones	212
2.8.5	Capturar una excepción	213
2.8.6	Manejar una excepción	214
2.8.7	Gestionar la salida del bloque de captura	216
2.8.8	Gestionar que no se produzcan excepciones	216

2.8.9	Gestor de contexto.	218
2.8.10	Programación asíncrona.	219
2.9	Otros	220
2.9.1	Gestionar imports	220
2.9.2	Compartir espacios de nombres	221
2.9.3	Funciones print, help, eval y exec	223

Capítulo 3-2

Declaraciones

1.	Variable.	225
1.1	¿Qué es una variable?	225
1.1.1	Contenido.	225
1.1.2	Continente	225
1.1.3	Formas de modificar una variable	227
1.2	Tipado dinámico	230
1.2.1	Asignación: recordatorio	230
1.2.2	Primitiva type y naturaleza del tipo	230
1.2.3	Características del tipado Python	231
1.3	Visibilidad	234
1.3.1	Espacio global	234
1.3.2	Noción de bloque	234
2.	Función.	238
2.1	Declaración	238
2.2	Parámetros.	239
2.2.1	Firma de una función.	239
2.2.2	Noción de argumento o de parámetro	240
2.2.3	Valor por defecto	240
2.2.4	Valor por defecto mutable.	242
2.2.5	Parámetros nombrados	243
2.2.6	Declaración de parámetros extensibles.	244
2.2.7	Paso de parámetros con asterisco	245
2.2.8	Firma universal.	246
2.2.9	Obligar a un parámetro a ser nombrado (keyword-only)	246
2.3	Forzar un argumento a ser posicional (positional-only).	248
2.3.1	Anotaciones/tipo hint/tipado estático	249
3.	Clase	252
3.1	Declaración	252
3.1.1	Firma.	252

3.1.2	Atributo	253
3.1.3	Método	253
3.1.4	Bloque local	254
3.2	Instanciación	255
3.2.1	Sintaxis	255
3.2.2	Relación entre la instancia y la clase	255
4.	Módulo	256
4.1	¿Para qué sirve un módulo?	256
4.2	Declaración	256
4.3	Instrucciones específicas	256
4.4	¿Cómo conocer el contenido de un módulo?	257
4.5	Compilación de los módulos	258

Capítulo 3-3 Modelo de objetos

1.	Todo es un objeto	261
1.1	Principios	261
1.1.1	Qué sentido dar a «objeto»	261
1.1.2	Adaptación de la teoría de objetos en Python	262
1.1.3	Generalidades	263
1.2	Clases	264
1.2.1	Introducción	264
1.2.2	Declaración imperativa de una clase	265
1.2.3	Instancia	265
1.2.4	Objeto en curso	267
1.2.5	Declaración por prototipo de una clase	267
1.2.6	Tuplas con nombre	269
1.3	Métodos	270
1.3.1	Declaración	270
1.3.2	Invocar al método	272
1.3.3	Métodos y atributos especiales	274
1.3.4	Constructor e inicializador	277
1.3.5	Gestión automática de atributos	278
1.3.6	Interés del paradigma orientado a objetos	279
1.3.7	Relación entre objetos	279
1.4	Herencia	280
1.4.1	Polimorfismo por subtipado	280
1.4.2	Sobrecarga de métodos	281

1.4.3	Sobrecarga de operadores	283
1.4.4	Polimorfismo paramétrico	284
1.4.5	Herencia múltiple	285
2.	Otras herramientas de la programación orientada a objetos	287
2.1	Principios	287
2.2	Interfaces	288
2.3	Atributos	290
2.4	Propiedades	293
2.5	Ubicaciones	295
2.6	Metaclases	296
2.7	Clases abstractas	299
2.8	Zope Component Architecture	301
2.8.1	Presentación	301
2.8.2	Instalación	302
2.8.3	Definir una interfaz y un componente	302
2.8.4	Otras funcionalidades	303
2.8.5	Ventajas de la ZCA	303
3.	Funciones principales y primitivas asociadas	304
3.1	Personalización	304
3.1.1	Clases	304
3.1.2	Instancias	306
3.1.3	Comparación	307
3.1.4	Evaluación booleana	307
3.1.5	Relaciones de herencia o de clase a instancia	307
3.2	Clases particulares	308
3.2.1	Iterador	308
3.2.2	Contenedores	310
3.2.3	Instancias similares a funciones	311
3.2.4	Recursos que hay que proteger	312
3.2.5	Tipos	312
3.2.6	Clases de datos	313

Capítulo 3-4

Números, booleanos y algoritmos aplicados

1.	Números	315
1.1	Tipos	315
1.1.1	Enteros	315
1.1.2	Reales	316

1.1.3	Cosas en común entre números enteros y reales	317
1.1.4	Métodos dedicados a los números enteros	318
1.1.5	Métodos dedicados a los números reales	319
1.1.6	Complejos	319
1.2	La consola Python, la calculadora por excelencia	320
1.2.1	Operadores matemáticos binarios	320
1.2.2	Operadores binarios particulares	321
1.2.3	Operadores matemáticos unarios	322
1.2.4	Redondeo	323
1.2.5	Operadores de comparación	325
1.2.6	Operaciones matemáticas n-arias	326
1.2.7	Funciones matemáticas usuales	328
1.3	Representaciones de un número	333
1.3.1	Representación decimal	333
1.3.2	Representación por un exponente	333
1.3.3	Representación por una fracción	333
1.3.4	Representación hexadecimal	334
1.3.5	Representación octal	335
1.3.6	Representación binaria	336
1.3.7	Operaciones binarias	336
1.3.8	Longitud de la representación en memoria de un entero	338
1.4	Conversiones	339
1.4.1	Conversión entre enteros y reales	339
1.4.2	Conversión entre reales y complejos	340
1.4.3	Conversión en un booleano	340
1.5	Trabajar con variables	341
1.5.1	Un número es inmutable	341
1.5.2	Modificar el valor de una variable	342
1.5.3	Operadores incrementales	342
1.6	Estadísticas	343
2.	Booleanos	344
2.1	El tipo booleano	344
2.1.1	Clase bool	344
2.1.2	Los dos objetos True y False	345
2.1.3	Diferencia entre el operador de igualdad y de identidad	345
2.2	Evaluación booleana	346
2.2.1	Método genérico	346
2.2.2	Objetos clásicos	346

Capítulo 3-5**Secuencias y algoritmos aplicados**

1. Presentación de los distintos tipos de secuencias.	347
1.1 Generalidades	347
1.2 Las listas.	348
1.3 Las n-tuplas	349
1.4 Conversión entre listas y n-tuplas	351
1.5 Cosas en común entre una lista y una n-tupla	351
2. Noción de iterador	352
3. Uso de índices y tramos.	355
3.1 Definición de índice de un objeto y sus ocurrencias.	355
3.2 Utilizar el índice para recorrer la secuencia	356
3.3 Encontrar las ocurrencias de un objeto y sus índices	357
3.4 Tamaño de una lista, contar ocurrencias.	358
3.5 Utilizar el índice para modificar o eliminar.	359
3.6 Iteración simple.	361
3.7 Presentación de la noción de tramos (slices)	364
3.8 Caso particular de la rama 2.x de Python	373
3.9 Uso básico de tramos	374
3.10 Uso avanzado de tramos.	375
4. Uso de operadores	377
4.1 Operador +	377
4.2 Operador *	378
4.3 Operador +=	381
4.4 Operador *=	382
4.5 Operador in	383
4.6 Operadores de comparación	384
5. Métodos de modificación	385
5.1 Agregar elementos a una lista y a una n-tupla	385
5.2 Eliminar un objeto de una lista y de una n-tupla	387
5.3 Soluciones alternativas para la modificación de n-tuplas.	391
5.4 Invertir una lista o una tupla	392
5.5 Ordenar una lista	393
6. Uso avanzado de listas	396
6.1 Operaciones de conjunto	396
6.2 Pivotar una secuencia	397

6.3	Iterar correctamente	398
6.4	Programación funcional	399
6.5	Recorrido de listas	401
6.6	Iteraciones avanzadas	403
6.7	Combinatoria	407
7.	Adaptar las listas a necesidades específicas	410
7.1	Lista de enteros	410
7.2	Presentación del tipo array	411
7.3	Utilizar una lista como una pila	413
7.4	Utilizar una lista como una fila de espera	414
7.5	Contenedor con mejor rendimiento	414
7.6	Utilizar las listas para representar matrices	415
7.7	Lista sin duplicados	416
8.	Otros tipos de datos	419

Capítulo 3-6 Conjuntos y algoritmos aplicados

1.	Presentación	423
1.1	Definición de un conjunto	423
1.2	Diferencias entre set y frozenset	424
1.3	Uso para eliminar valores duplicados de las listas	425
1.4	Agregar una relación de orden	425
2.	Operaciones sobre conjuntos	426
2.1	Operadores para un conjunto a partir de otros dos	426
2.2	Operadores para modificar un conjunto a partir de otro	427
2.3	Métodos equivalentes a la creación o modificación de conjuntos	428
2.4	Métodos de comparación de conjuntos	428
2.5	Ejemplos de uso poco clásicos	430
3.	Métodos de modificación de un conjunto	433
3.1	Agregar un elemento	433
3.2	Eliminar un elemento	434
3.3	Vaciar un conjunto	434
3.4	Duplicar un elemento	434
3.5	Sacar un valor de un conjunto	435
3.6	Utilizar un conjunto como un almacén de objetos	436
3.7	Algorítmica avanzada: resolución del problema de las n reinas	439

Capítulo 3-7

Cadenas de caracteres y algoritmos aplicados

1.	Presentación	443
1.1	Definición	443
1.2	Vocabulario	444
1.3	Especificidades de la rama 2.x	445
1.4	Cambios aportados por la rama 3.x	446
1.5	Cadena de caracteres como secuencia de caracteres	448
1.6	Caracteres	450
1.7	Operadores de comparación	452
2.	Dar formato a cadenas de caracteres	454
2.1	Operador módulo	454
2.2	Métodos para dar formato al conjunto de la cadena	459
2.3	Nuevo método para dar formato a variables en una cadena	462
2.4	Literales formateados	465
3.	Operaciones de conjunto.	466
3.1	Secuenciación de cadenas	466
3.2	Operaciones sobre mayúsculas y minúsculas	468
3.3	Búsqueda en una cadena de caracteres	469
3.4	Información sobre los caracteres	470
4.	Problemáticas relativas a la codificación	472
4.1	Codificación por defecto.	472
4.2	Codificación del sistema	472
4.3	Unicode, referencia absoluta	472
4.4	Otras codificaciones	473
4.5	Puntos entre el Unicode y el resto del mundo.	474
4.6	Volver a Unicode.	475
5.	Manipulaciones de bajo nivel avanzadas	476
5.1	Operaciones para contar	476
5.2	Una cadena de caracteres vista como una lista	477
5.3	Una cadena de caracteres vista como un conjunto de caracteres.	478
6.	Representación en memoria	478
6.1	Presentación del tipo bytes.	478
6.2	Vínculo con las cadenas de caracteres	479
6.3	Presentación del tipo bytearray	480
6.4	Gestión de un juego de caracteres	482

Capítulo 3-8

Diccionarios y algoritmos aplicados

1. Presentación	489
1.1 Definición	489
1.2 Evolución y diferencias entre las ramas 2.x y 3.x	490
1.3 Vistas de diccionarios	491
1.4 Instanciación	493
1.5 Recorrer un diccionario	494
2. Manipular un diccionario	494
2.1 Recuperar un valor de un diccionario	494
2.2 Modificar los valores de un diccionario	495
2.3 Eliminar una entrada de un diccionario	496
2.4 Duplicar un diccionario	497
2.5 Utilizar un diccionario como un agregador de datos	497
2.6 Métodos de iteración	498
3. Uso avanzado de diccionarios	499
3.1 Agregar una relación de orden	499
3.2 Algorítmicas clásicas	500
3.3 Adaptar los diccionarios a necesidades específicas	502
3.4 Representación universal de datos	504

Capítulo 3-9

Datos temporales y algoritmos aplicados

1. Gestionar una fecha del calendario	507
1.1 Noción de fecha del calendario	507
1.2 Trabajar con una fecha	508
1.3 Consideraciones astronómicas	509
1.4 Consideraciones históricas	509
1.5 Consideraciones técnicas	509
1.6 Representación textual	510
2. Gestionar un horario o un momento de la jornada	511
2.1 Noción de instante	511
2.2 Noción de huso horario	513
2.3 Representación textual	513

3.	Gestionar un instante absoluto.	514
3.1	Noción de instante absoluto	514
3.2	Relación con las nociones anteriores	514
3.3	Representación textual.	516
3.4	Gestión de los husos horarios.	517
3.5	Crear una fecha a partir de una representación textual	517
4.	Gestionar una diferencia entre dos fechas o instantes	517
4.1	Noción de diferencia y de resolución	517
4.2	Consideraciones técnicas	519
4.3	Uso con fechas del calendario.	520
4.4	Uso con horarios	520
4.5	Uso con fechas absolutas	520
4.6	El segundo como unidad básica	520
4.7	Precisión al nanosegundo	521
5.	Especificidades de los husos horarios	521
6.	Problemáticas de bajo nivel.	522
6.1	Timestamp y struct_time.	522
6.2	Medidas de rendimiento	523
7.	Uso del calendario	526
7.1	Presentación del módulo calendar	526
7.2	Funciones esenciales del calendario	530

Parte 4: Características

Capítulo 4-1

Manipulación de datos

1.	Manipular los archivos	533
1.1	Abrir un archivo	533
1.2	Leer un archivo	534
1.3	Escribir un archivo	535
1.4	Comparar dos archivos.	536
2.	Herramienta de copia de seguridad.	538
3.	Leer un archivo de configuración	538
4.	Formato de exportación/importación	539
4.1	CSV	539
4.1.1	Explotar un archivo CSV.	540
4.1.2	Generación de un archivo CSV	544

4.2	JSON	546
4.3	Base64	549
4.4	Pickle	549
5.	Comprimir y descomprimir un archivo	552
5.1	Tarfile	552
5.2	Gzip	554
5.3	BZ2	554
5.4	Zipfile	555
5.5	Interfaz de alto nivel	557
6.	Herramientas de manipulación de datos	558
6.1	Generar números aleatorios	558
6.2	Expresiones regulares	559
7.	Criptografía ligera	563
7.1	Número aleatorio seguro	563
7.2	Funciones de cifrado	563
7.3	Código de autenticación del mensaje	565
7.4	Huella de archivo	566
7.5	Esteganografía	567
7.6	Comunicación segura entre aplicaciones	570

Capítulo 4-2

Bases de datos

1.	Introducción	573
2.	Acceso a una base de datos relacional	573
2.1	Punto de entrada	573
2.2	MySQL	574
2.3	PostgreSQL	579
2.4	SQLite	581
2.5	Oracle	582
3.	Uso de un ORM	582
3.1	¿Qué es un ORM?	582
3.2	ORM propuestos por Python	582
3.3	SQLAlchemy	583
3.3.1	Introspección en una tabla existente	583
3.3.2	Manipular datos en una tabla existente	586
3.3.3	Describir una base de datos mediante el código	590

4.	Otras bases de datos	591
4.1	CSV	591
4.2	NoSQL	597
4.3	Base de datos orientada a objetos: ZODB	597
4.4	Base de datos orientada al grafo: Neo4j	602
4.5	Base de datos de tipo clave-valor: Redis	603
4.6	Bases de datos orientadas a documentos: CouchDB y MongoDB ..	605
4.7	Bases de datos nativas XML: BaseX, eXist	606
4.8	Cassandra	607
4.9	Bases de datos orientadas a columnas: HBase	607
4.10	Big Data: el ecosistema Hadoop	609
5.	LDAP	611
5.1	Protocolo	611
5.2	Servidores	612
5.3	Terminología	612
5.4	Instalación	612
5.5	Abrir una conexión a un servidor	613
5.6	Realizar una búsqueda	614
5.7	Síncrono vs asíncrono	615
5.8	Conexiones seguras	616

Parte 5: Puesta en práctica

Capítulo 5-1

Crear un entorno de trabajo en 10 minutos

1.	Descripción de la aplicación para construir	617
2.	Contenedores	618
2.1	Portainer	618
2.2	Base de datos	619
3.	Crear su contenedor Docker	621
4.	Instalar sus bibliotecas Python	623

Capítulo 5-2

Crear una aplicación web en 30 minutos

1. Descripción de la aplicación que se va a construir.	625
2. Implementación.	626
2.1 Aislar el entorno	626
2.2 Creación del proyecto.	627
2.3 Configuración	627
2.4 Primeros ensayos.	629
3. Realizar la aplicación.	630
3.1 Modelos.	630
3.2 Plantillas	632
3.3 Vistas	634
4. Para ir más allá.	638

Capítulo 5-3

Crear una aplicación de consola en 10 minutos

1. Objetivo	639
2. Registrar el script.	640
3. Creación de los datos.	640
4. Parser de argumentos	641

Capítulo 5-4

Crear una aplicación gráfica en 20 minutos

1. Objetivo	643
1.1 Funcional.	643
1.2 Técnico	643
2. Breve presentación de Gtk y algunos trucos	644
2.1 Presentación	644
2.2 Trucos	644
3. Iniciar el programa.	645
4. Interfaz gráfica con Glade.	648
5. Crear el componente gráfico.	650
6. Controlador	652

7.	Otras librerías gráficas	653
7.1	TkInter	653
7.2	wxPython	653
7.3	PyQt	653
7.4	PySide	654
7.5	Otras	654

Capítulo 5-5

Crear un juego en 30 minutos con PyGame

1.	Presentación de PyGame	655
2.	Construcción de un juego Tetris	656
2.1	Presentación del juego	656
2.2	Presentación de la problemática	657
2.3	Creación de constantes	657

Anexos

1.	Objetos mutables y no mutables	669
2.	Tabla Unicode	672
2.1	Script	672
3.	Bytes	672
3.1	Script	672
3.2	Resultado	673
4.	Guía de migración a Python 3	676
5.	Cómo depurar	677
6.	Cómo probar el rendimiento	679

Índice	681
------------------	-----

Podrá descargar algunos elementos de este libro en la página web de Ediciones ENI: <http://www.ediciones-eni.com>.
Escriba la referencia ENI del libro **EIT23PYT** en la zona de búsqueda y valide.
Haga clic en el título y después en el botón de descarga.

Vista previa

Parte 1

Tratamiento de los datos

Capítulo 1.1

Patrones de diseño

1. Definición	15
1.1 Situación respecto a la noción de objeto	15
1.2 Organización del capítulo	16
1.3 Situación respecto a otros conceptos.	16
2. Patrones de creación	17
2.1 Singleton (Instancia única)	17
2.2 Fábrica	19
2.3 Fábrica abstracta	22
2.4 Constructor	24
2.5 Prototipo	27
3. Patrones de estructuración	30
3.1 Adaptador	30
3.2 Puente	34
3.3 Composite	37
3.4 Decorador	40
3.5 Fachada	43
3.6 Peso mosca	45
3.7 Proxy	46
4. Patrones de comportamiento	49
4.1 Cadena de responsabilidad	49
4.2 Solicitudes	50

4.3	Iterador	52
4.4	Memento (recuerdo)	55
4.5	Visitante	56
4.6	Observador	57
4.7	Estrategia	59
4.8	Función de callback	60
5.	ZCA	61
5.1	Consideraciones	61
5.2	Adaptador	61
5.3	Utilidad	63
5.4	Fábrica	64
5.5	Para ir más allá	65

Capítulo 1.2

XML

1.	XML y las tecnologías relacionadas	67
1.1	Definición de XML, terminología asociada	67
1.2	Noción de esquema	68
1.3	Ventajas e inconvenientes de XML	69
1.4	Distintas maneras de recorrer un archivo XML	71
1.5	Módulos Python dedicados a XML	71
2.	Validar un documento XML	72
2.1	Documento XML	72
2.2	Esquema DTD	73
2.3	Esquema XSD	74
2.4	Esquema RNG (RelaxNG)	75
2.5	Schematron	75
3.	DOM	76
3.1	Lectura	76
3.2	Escritura	77
4.	SAX	79
4.1	Soporte de SAX en lxml	79
4.2	API SAX ligera	80
5.	XPath	81

6.	XSLT	84
7.	El caso concreto de los archivos HTML	85
7.1	Problemática	85
7.2	Analizar sintácticamente un archivo HTML según DOM	86
7.3	Analizar sintácticamente un archivo HTML según SAX	88

Capítulo 1.3

Generación de contenido

1.	PDF	91
1.1	Presentación	91
1.1.1	Formato PDF	91
1.1.2	Ventajas	91
1.1.3	Inconvenientes	92
1.1.4	Presentación de la librería libre	92
1.2	Bajo nivel	92
1.2.1	Librería de datos	92
1.2.2	Canvas (Lienzo)	95
1.3	Alto nivel	96
1.3.1	Estilos	96
1.3.2	Flujo de datos	98
1.3.3	Crear un elemento visual	101
1.3.4	Plantilla de página	101
1.3.5	Página que contiene varias zonas	103
2.	Generar un documento PDF a partir de una página HTML	105
2.1	Presentación	105
2.2	Instalación	105
2.3	Uso	106
3.	OpenDocument	106
3.1	Presentación	106
3.2	ezodf2	107
3.2.1	Instalación	107
3.2.2	Texto OpenDocument	107
3.2.3	Hoja de cálculo OpenDocument	107
3.2.4	Ir más allá	108
3.3	Alternativas	108

4 --- Python 3

Tratamiento de datos y técnicas de programación

4.	Trabajar con imágenes.	108
4.1	Representación informática de una imagen	108
4.2	Presentación de Pillow	109
4.3	Formatos de imágenes matriciales	111
4.4	Recuperar información de una imagen	113
4.5	Operaciones generales sobre una imagen	114
4.6	Trabajar con las capas o los píxeles	117
5.	Archivos de configuración.	119
5.1	Formato YAML	120
5.2	Formato TOML.	121

Capítulo 1.4

Calidad

1.	Desarrollo guiado por pruebas.	123
1.1	Pruebas unitarias.	123
1.1.1	Principios	123
1.1.2	Interpretar	124
1.1.3	Cobertura	125
1.1.4	Herramientas	126
1.1.5	Otras herramientas	128
1.2	Pruebas de no regresión.	128
1.2.1	Acciones de desarrollo	128
1.2.2	Gestionar las anomalías detectadas por el cliente	129
1.3	Pruebas funcionales.	130
1.4	Pruebas de rendimiento	131
1.5	Pruebas sintácticas	134
1.6	Integración continua.	135
2.	Programación dirigida por la documentación	136
2.1	Documentación interna	136
2.1.1	Destinada a los desarrolladores	136
2.1.2	Destinada a los usuarios	137
2.2	Documentación externa	138
2.2.1	Presentación	138
2.2.2	Inicio rápido	138
2.2.3	Resultado	141

3.	Optimizar.	141
3.1	Medir la calidad.	141
3.2	Herramientas de depuración	143
3.3	Herramientas de perfilado	145
3.4	Reglas para optimizar.	146
3.4.1	¿Por qué optimizar?.	146
3.4.2	Reglas generales	147
3.4.3	Optimizar un algoritmo	148
3.4.4	Optimizar el uso de la memoria	159

Parte 2

Programación del sistema, la red y científica

Capítulo 2.1

Programación de sistema

1.	Presentación.	161
1.1	Definición	161
1.2	Objetivos del capítulo.	162
2.	Comprender su sistema operativo	162
2.1	Advertencia.	162
2.2	Sistema operativo	162
2.3	Proceso actual	163
2.4	Usuarios y grupos	164
2.5	Contraseñas y autenticación	167
2.6	Constantes para el sistema de archivos.	169
2.7	Administrar las rutas	169
2.8	Herramientas	170
2.8.1	Comparar dos archivos	170
2.8.2	Herramienta de copia de seguridad	173
2.8.3	Leer un archivo de configuración	173
2.8.4	Pickle.	174

2.9	Comprimir y descomprimir un archivo	177
2.9.1	Tarfile	177
2.9.2	Gzip.	180
2.9.3	Bz2	180
2.9.4	LZMA	181
2.9.5	Zipfile	181
2.9.6	Interfaz de alto nivel	183
3.	Gestionar un archivo	185
3.1	Cambiar los permisos de un archivo	185
3.2	Cambiar de propietario o de grupo	187
3.3	Recuperar información de un archivo	188
3.4	Eliminar un archivo.	189
4.	Alternativas sencillas a los comandos bash usuales.	189
4.1	Directorios	189
4.2	Archivos	193
4.3	Módulo de alto nivel	194
4.4	Buscar un archivo	196
5.	Ejecutar comandos externos	197
5.1	Ejecutar y mostrar el resultado	197
5.2	Ejecutar y recuperar el resultado	198
5.3	Para Python 3.5 y superior	199
6.	Sistema IHM	200
6.1	Presentación.	200
6.2	Trabajar con argumentos	201
6.3	Cerrar el programa	206
6.4	Entrada estándar	207
6.5	Salida estándar	208
6.6	Salida de error	209
6.7	Tamaño del terminal	210
7.	Curses	211
7.1	Presentación.	211
7.2	Administrar el arranque y la parada.	211
7.3	Mostrar texto	213
7.4	Gestionar atributos	213
7.5	Gestionar los colores	214
7.6	Gestionar los eventos	214

8.	Acceso a los periféricos	214
8.1	Introducción	215
8.2	Ver las particiones.	215
8.3	Ver las llaves USB	216
8.4	Lista de la información disponible	217
8.5	Detectar la conexión de una clave USB.	217
8.6	Comunicar vía un puerto en serie	218
8.7	Comunicar vía un puerto USB.	220

Capítulo 2.2

Programación de red

1.	Escribir un servidor y un cliente	223
1.1	Uso de un socket TCP	223
1.2	Uso de un socket UDP	227
1.3	Crear un servidor TCP	230
1.4	Crear un servidor UDP	233
1.5	Ir más allá en el tema	233
2.	Utilizar un protocolo estándar	235
2.1	Cliente HTTP	235
2.2	Servidor HTTP	236
2.3	Proxy	240
2.4	Cookies	241
2.5	FTP y SFTP	241
2.6	SSH	244
2.7	POP y POPS.	246
2.8	IMAP y IMAPS	247
2.9	SMTP y SMTPS	249
2.10	NNTP	254
2.11	IRC.	255
3.	Wake-on-LAN (WoL)	259
3.1	Requisitos previos.	259
3.2	Implementación	259

Capítulo 2.3

Programación web

1. Servicios web	261
1.1 REST	261
1.2 SOAP	263
1.3 Pyro	265
2. Cliente web	266
3. Extraer datos de la Web (Web scrapping)	271

Capítulo 2.4

Programación científica

1. Cálculo científico	275
1.1 Presentación.	275
1.2 Python, una alternativa libre y creíble.	276
1.3 Descripción general de algunas librerías	276
2. Tablas multidimensionales	277
2.1 Crear.	277
2.2 Determinar la composición de una tabla.	279
2.3 Generadores de tablas	279
2.4 Operaciones básicas	281
2.5 Operador corchete.	284
3. Matrices	287
4. Generar gráficos	288
4.1 Sintaxis MATLAB.	288
4.2 Sintaxis de objeto	290
4.3 Formatear	290
4.4 Gráficos 3D	295
5. Introducción a Pandas	299
5.1 Presentación.	299
5.2 Series	300
5.3 Dataframes (Estructura de datos)	305

Parte 3

Programación concurrente

Capítulo 3.1

Iniciación a la programación concurrente

1. Noción de rendimiento	315
1.1 Programación bloqueante	315
1.2 Utilizar recursos de CPU	317
1.3 Organización de la aplicación	318
2. Terminología	319
2.1 Proceso	319
2.2 Hilo de ejecución	320
2.3 Programación no bloqueante	321
2.4 Noción de GIL	321
3. Presentación de los paradigmas	322
3.1 Programación asíncrona	322
3.2 Programación en paralelo	323
3.3 Programación distribuida	324
3.4 Programación concurrente	325

Capítulo 3.2

Programación asíncrona: iniciación

1. Utilidad de la programación asíncrona	329
2. Introducción a la asincronía	330
2.1 Noción de subrutina	330
2.2 Ejecutar una subrutina	332
2.3 Precisiones sobre la palabra clave await	333
2.4 Ejecutar varias subrutinas en serie	335
2.5 Tareas asíncronas	336
2.6 Ejecutar las tareas asíncronas de forma concurrente	337
2.7 Noción de awaitable	338
2.8 Precisiones sobre asyncio.sleep	338

3.	Elementos de gramática.	338
3.1	Administrador de contexto asíncrono.	338
3.2	Generadores asíncronos	339
3.3	Comprensión asíncronas	340
3.4	Iteración asíncrona	340
4.	Nociones avanzadas.	340
4.1	Introspección.	340
4.2	Gestionar el resultado o la excepción.	344
4.3	Cancelar una tarea	346
4.4	Proteger contra una cancelación	348
4.5	Timeouts	348
4.6	Gestión global afinada de la espera	349
4.7	Módulo contextvars	352
5.	Bucle orientado a eventos asíncrono.	356
5.1	Gestionar bucles	356
5.2	Depurar	357
5.3	Noción de futuro.	359
5.4	Cancelar.	362
5.5	Gestionar excepciones.	363
6.	Utilizar bucles orientados a eventos.	368
6.1	Utilizar las funciones de retrollamada (callbacks)	368
6.2	Planificar las llamadas a las funciones clásicas	371
6.3	Utilizar señales	373
6.4	Sincronizar.	375
6.5	Comunicación entre subrutinas.	381
6.6	Ejemplo: lectura de la entrada estándar.	385
6.7	Programación en modo paralelo y asíncrono.	386
6.8	Ejecutar código bloqueante	386
7.	La asincronía según las versiones de Python	387
7.1	Python 3.7	387
7.2	Python 3.6	389
7.3	Python 3.5	389
7.4	Python 3.4	389
7.5	Python 3.3 e inferior	390
7.6	Python 2.7	390

8. Caso concreto	391
8.1 Ejemplo de trabajo	391
9. Ejemplo adaptado usando generadores	394
9.1 Descargar en modo asíncrono	396
9.2 Análisis sintáctico en modo asíncrono	397
9.3 Hacer varios tratamientos en modo asíncrono	398
9.4 Utilizar un ejecutor	399
9.5 Para ir más lejos	400

Capítulo 3.3 Programación asíncrona: avanzada

1. Transportes y protocolos	401
1.1 Introducción	401
1.2 Transportes	402
1.3 Protocolos	406
1.4 Implementación	407
1.5 Ejemplo para el protocolo TCP	409
1.6 Ejemplo para el protocolo SSL	418
1.7 Ejemplo para el protocolo UDP	422
1.8 Tratar otros tipos de protocolos de red	426
1.9 Ejemplo para Subprocess	426
2. Flujo	434
2.1 Introducción	434
2.2 Ejemplo con el protocolo TCP	434

Capítulo 3.4 Programación asíncrona: alternativas

1. Gevent	437
1.1 Introducción	437
1.2 DNS	438
1.3 Llamadas de sistema	438
1.4 Programación asíncrona	439

2.	Twisted	441
2.1	Introducción	441
2.2	Cliente/servidor TCP	441
2.3	Servidor/cliente UDP	443
2.4	AMP	445
2.5	Otros protocolos	448
3.	Trollius	449
4.	HTTP asíncrono con aiohttp	451
4.1	Lado servidor	451
4.2	Lado cliente	453

Capítulo 3.5

Programación en paralelo

1.	Uso de un hilo de ejecución	455
1.1	Gestionar un hilo de ejecución	455
1.1.1	Presentación	455
1.1.2	Crear	456
1.2	Gestionar varios hilos de ejecución	459
1.2.1	Lanzar y controlar	459
1.2.2	Oportunidad de utilizar un hilo de ejecución	461
1.3	Resolver los problemas relacionados	463
1.3.1	Sincronizar	463
1.3.2	Sincronizar con condición	466
1.3.3	Semáforo	469
2.	Uso de proceso	471
2.1	Gestionar un proceso	471
2.1.1	Presentación	471
2.1.2	Crear	471
2.2	Gestionar varios procesos	474
2.2.1	Sincronizar	474
2.2.2	Paralelizar un trabajo	475
2.3	Resolver los problemas relacionados	477
2.3.1	Comunicación interproceso	477
2.4	Compartir datos entre procesos	478
2.5	Oportunidad de utilizar los procesos	480

2.6	Demonio	480
3.	Ejecutar de forma asíncrona	482
3.1	Introducción	482
3.2	Presentación	483

Capítulo 3.6

Programación distribuida

1.	Definiciones	491
2.	ØMQ	492
2.1	Presentación general	492
2.2	Pair	493
2.3	Cliente/servidor	495
2.4	Publish/subscribe	496
2.5	PUSH/PULL	501
2.6	Patrón pipeline	502
3.	AMQP con RabbitMQ	504
3.1	Instalar	505
3.2	Configurar	505
3.3	Introducción	505
3.4	Nociones importantes	508
3.5	Publish/subscribe	509
3.6	Enrutamiento y temas	511
4.	Kafka	513
4.1	Presentación	513
4.2	Principios generales	513
4.3	Cliente Kafka	514
4.4	Faust	515
5.	Celery	516
5.1	Presentación	516
5.2	¿En qué caso utilizar Celery?	516
5.3	Instalar	517
5.4	Configurar	517
5.5	Utilizar	519
5.6	Monitorizar	521

6. Crossbar	522
6.1 Presentación.....	522
6.2 WebSocket.....	522
6.3 Publish/subscribe	529
Índice	533