

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/352050747>

PYTHON APLICADO

Book · March 2021

CITATIONS

0

READS

3,694

1 author:



Eugenia Bahit

Bahit & Bahit Ltd

25 PUBLICATIONS 1 CITATION

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Python for beginners [View project](#)



SACRED [View project](#)

PYTHON APLICADO

EUGENIA BAHIT

Informática Teórica[†]
Elida Bahit Research Centre



EBRC Publisher

Londres • Madrid • Buenos Aires • México • Bogotá
2021

[†] Directora del Elida Bahit Research Centre de Reino Unido. Contacto: www.elidabahit.co.uk

PYTHON APLICADO
© 2020 EUGENIA BAHIT

Excepto donde se indique lo contrario, el contenido de este libro está distribuido bajo una licencia Creative Commons Atribución No Comercial Sin Derivadas 4.0 Internacional (CC BY-NC-ND 4.0). La copia y distribución sin fines de lucro se encuentran permitidas. Texto completo de la licencia:
<https://creativecommons.org/licenses/by-nc-nd/4.0/deed.es>

ISBN: 978-1-8381901-2-5
e-ISBN: 978-1-8381901-3-2

Título original: Python para principiantes, Edición 2020
Publicado en Reino Unido por EBRC Publisher
71-75 Shelton Street, Covent Garden, London, WC2H 9JQ
www.elidabahit.co.uk

Editora: Eugenia Bahit

Diseño interior: Amanda Kraft
Diseño de portada: Julia Davies

Impreso en:

Argentina por Docuprint S.A.
para distribución exclusiva en el territorio argentino, Chile, Paraguay y Uruguay.

Colombia, por Bubok Publishing S.L.
para distribución exclusiva en territorio colombiano.

España, por Bubok Publishing S.L.
para distribución exclusiva en territorio español.

México, por Bubok Publishing S.L.
para distribución exclusiva en territorio mexicano.

Estados Unidos, por Lulu Press, Inc.
para distribución en América Latina (excepto Argentina), Reino Unido, Unión Europea, Asia, África y Oceanía.

*A quienes construyen sus sueños con
pasión y se esfuerzan para alcanzar
sus metas.*

Índice de contenidos

Prólogo: entender la diferencia entre programar y codificar.....	7
1. Primer acercamiento al Scripting.....	11
Convertir un script en comando del sistema.....	12
2. Acerca de Python.....	13
Glosario.....	13
3. Elementos del Lenguaje.....	15
Variables.....	15
Entrada y salida.....	16
Tipos de datos.....	17
Codificación de caracteres (encoding).....	18
Operadores Aritméticos.....	19
Comentarios.....	19
4. Tipos de datos complejos.....	20
Tuplas.....	21
Listas.....	21
Diccionarios.....	22
5. Estructuras de Control de Flujo.....	23
Sangrado.....	23
Estructuras de control de flujo condicionales.....	24
Operadores lógicos.....	25
Estructuras de control iterativas.....	26
Bucle while.....	27
Bucle for.....	29
6. Funciones.....	30
Funciones definidas por el usuario.....	30
Sobre los parámetros.....	31
Parámetros por omisión.....	32
Claves como argumentos.....	33
Parámetros arbitrarios.....	33
Desempaquetado de parámetros.....	34
Llamadas recursivas y de retorno.....	35
Sobre la finalidad de las funciones.....	36
Conceptos avanzados sobre funciones.....	36
Lambdas.....	36
Clausuras (closures).....	37
Envolturas (wrappers) y decoradores.....	39
Funcionamiento de las envolturas y decoradores.....	40
7. Importación de módulos.....	42
8. Manipulación de cadenas de texto.....	45

Inyección de variables.....	45
Métodos de formato.....	47
Convertir a mayúscula la primera letra.....	47
Convertir una cadena a minúsculas.....	47
Convertir una cadena a mayúsculas.....	47
Convertir mayúsculas a minúsculas y viceversa.....	47
Convertir una cadena en Formato Título.....	48
Centrar un texto.....	48
Alinear texto a la izquierda.....	48
Alinear texto a la derecha.....	49
Rellenar un texto anteponiendo ceros.....	49
Métodos de Búsqueda.....	49
Contar cantidad de apariciones de un fragmento de texto.....	49
Buscar un fragmento de texto dentro de una cadena.....	50
Métodos de Validación.....	50
Saber si una cadena comienza por un texto determinado.....	50
Saber si una cadena finaliza con un texto determinado.....	50
Saber si una cadena es alfanumérica.....	51
Saber si una cadena es alfabética.....	51
Saber si una cadena es numérica.....	51
Saber si una cadena contiene solo minúsculas.....	52
Saber si una cadena contiene solo mayúsculas.....	52
Saber si una cadena contiene solo espacios en blanco.....	53
Saber si una cadena tiene formato tipo título.....	53
Métodos de Sustitución.....	53
Dar formato a una cadena, sustituyendo texto dinámicamente.....	53
Reemplazar texto en una cadena.....	54
Eliminar caracteres a la izquierda y derecha de una cadena.....	54
Eliminar caracteres a la izquierda de una cadena.....	55
Eliminar caracteres a la derecha de una cadena.....	55
Métodos de unión y división.....	55
Unir una cadena de forma iterativa.....	55
Partir una cadena en tres partes, utilizando un separador.....	56
Partir una cadena en varias partes, utilizando un separador.....	56
Partir una cadena en en líneas.....	56
9. Manipulación de listas y tuplas.....	57
Métodos de agregado.....	57
Agregar un elemento al final de la lista.....	57
Agregar varios elementos al final de la lista.....	57
Agregar un elemento en una posición determinada.....	58
Métodos de eliminación.....	58
Eliminar el último elemento de la lista.....	58

Eliminar un elemento por su índice.....	58
Eliminar un elemento por su valor.....	59
Métodos de orden.....	59
Ordenar una lista en reversa (invertir orden).....	59
Ordenar una lista en forma ascendente.....	59
Ordenar una lista en forma descendente.....	59
Métodos de búsqueda.....	60
Contar cantidad de apariciones elementos.....	60
Obtener número de índice.....	60
Anexo sobre listas y tuplas.....	60
Conversión de tipos.....	60
Concatenación de colecciones.....	61
Valor máximo y mínimo.....	61
Contar elementos.....	62
10. Manipulación de diccionarios.....	62
Métodos de eliminación.....	62
Vaciar un diccionario.....	62
Métodos de agregado y creación.....	63
Copiar un diccionario.....	63
Crear un nuevo diccionario desde las claves de una secuencia.....	63
Concatenar diccionarios.....	64
Establecer una clave y valor por defecto.....	64
Métodos de retorno.....	65
Obtener el valor de una clave.....	65
Saber si una clave existe en el diccionario.....	65
Obtener las claves y valores de un diccionario.....	65
Obtener las claves de un diccionario.....	67
Obtener los valores de un diccionario.....	67
Obtener la cantidad de elementos de un diccionario.....	68
11. Manejo y manipulación de archivos.....	68
Modos de Apertura de un archivo.....	68
Algunos métodos del Objeto File.....	70
Acceso a archivos mediante la estructura with.....	71
12. Manejo de archivos CSV.....	71
Algunos ejemplos de archivos CSV.....	72
Trabajar con archivos CSV desde Python.....	74
Lectura de archivos CSV.....	74
Escritura de archivos CSV.....	76
13. Manipulación avanzada de cadenas de texto.....	78
Expresiones regulares en Python.....	79
14. Creando menús de opciones.....	81
Creación de un menú de opciones básico.....	81

Creación de un menú de opciones con argparse.....	82
Paso 1: Importación del módulo.....	83
Paso 2: Construcción de un objeto ArgumentParser.....	83
Paso 3: Agregado de argumentos y configuración.....	83
Paso 4: Generación del análisis (parsing) de argumentos.....	86
15. Generación de registros de sistema.....	88
Principales elementos del módulo logging.....	89
Obtención de argumentos por línea de comandos con argv.....	93
Captura básica de excepciones con try y except.....	93
16. Módulos del sistema (os, sys y subprocess).....	95
El módulo OS.....	95
Variables de entorno: os.environ.....	97
Ejecución simplificada de comandos del sistema.....	98
Ejecución de comandos del sistema mediante Popen y shlex.split.....	99
Capturar la salida estándar y los errores.....	100
Emplear la salida de un comando como entrada de otro.....	101
Variables y funciones del módulo sys.....	102
17. Conexiones remotas (HTTP, FTP y SSH).....	105
Conexiones remotas vía HTTP y HTTPS.....	105
Conexiones remotas vía FTP.....	107
Solicitando la contraseña con getpass.....	109
Conexiones SSH con Paramiko.....	109
Requisitos previos.....	109
Uso de Paramiko.....	110
18. Bibliotecas para manejo avanzado de archivos en GNU/Linux.....	112
Compresión y descompresión de archivos con Itarfile y zipfile.....	112
La biblioteca tarfile.....	112
La biblioteca zipfile.....	113
Manejo de archivos temporales con la biblioteca tempfile.....	114
Lectoescritura de archivos temporales.....	114
Búsqueda de archivos con las bibliotecas glob y fnmatch.....	116
19. Probabilidad y Estadística con Python.....	117
Funciones estadísticas básicas (len, sum, max, min).....	117
Probabilidad de sucesos simples y compuestos mutuamente excluyentes.	117
Espacio muestral.....	117
Sucesos simples y compuestos.....	118
Asignación de probabilidades.....	118
Sucesos simples mutuamente excluyentes.....	119
Sucesos compuestos por sucesos simples mutuamente excluyentes....	119
Funciones.....	121
Probabilidad condicional en Python.....	121
Funciones.....	122

Sucesos dependientes.....	123
Teoría de conjuntos en Python.....	125
Sucesos independientes.....	126
Teorema de Bayes en Python.....	126
Teorema de Bayes y probabilidad de causas.....	126
Datos: caso práctico.....	127
Análisis.....	127
Procedimiento.....	129
Funciones.....	134
Bibliografía complementaria.....	134
20. Estadística descriptiva con Python.....	135
Estadística poblacional y muestral.....	135
Medidas descriptivas de tendencia central.....	135
Medidas descriptivas de dispersión.....	136
Cálculos de dispersión.....	136
Frecuencia estadística.....	138
Frecuencia absoluta.....	138
Frecuencia relativa.....	140
Frecuencias acumuladas.....	141
21. Python como CGI para aplicaciones Web.....	142
Entender la interfaz CGI.....	143
Entender el servidor HTTP de Apache.....	144
Montar un Virtual Host localmente.....	147
Instalación y configuración de Apache.....	147
Habilitación del módulo cgi.....	148
Definición de un nombre de host nuevo.....	149
Creación de la estructura de directorios.....	150
Creación del Virtual Host.....	151
Habilitación del nuevo Virtual Host.....	151
Reiniciar Apache.....	152
Probando la nueva Web.....	152
Separar el HTML del código Python.....	153
Envío de correo electrónico.....	155
Métodos GET y POST de HTTP.....	158
Recibiendo y analizando solicitudes por GET.....	158
El método POST: trabajar con datos enviados desde un formulario.....	160
Carga de archivos con Python.....	161
Consideraciones sobre la seguridad.....	161
Servir archivos estáticos con Python.....	162
Obtener el tipo MIME de un archivo.....	167
Codificar un archivo en Base 64.....	168
22. Conexiones a bases de datos con MySQL® y MariaDB.....	169

Configuración de MariaDB.....	170
Trabajando con MariaDB y MySQL® desde Python.....	172
Seguridad: prevención de inyecciones SQL.....	178
Función para automatizar consultas SQL.....	180
23. Programación orientada a objetos con Python.....	181
Breve introducción a la programación orientada a objetos.....	182
Elementos y características de la programación orientada a objetos.....	183
Clases.....	183
Métodos y propiedades.....	183
Objetos.....	184
Polimorfismo.....	184
Encapsulado.....	185
Herencia.....	185
Composición.....	186
Visibilidad y Ocultación.....	187
Sobre el uso de <code>self</code> en Python.....	190
Artilugios de la programación orientada a objetos.....	190
El método constructor.....	190
Recorrido de propiedades.....	191
Patrón de diseño compuesto y agregación.....	191
24. Pruebas unitarias.....	193
Doctest.....	193
Unittest.....	195
Métodos Assert.....	195
Descubriendo pruebas.....	197

Prólogo: entender la diferencia entre programar y codificar

En el ámbito de la ingeniería de software, un **problema** es un requerimiento concreto que necesita ser resuelto mediante un programa informático, mientras que su **análisis** consiste en entender qué se necesita hacer para determinar cómo hacerlo, mediante qué operaciones, con cuáles valores de entrada y para arrojar cuáles valores de salida. Los **algoritmos** constituyen el paso a paso de la solución del problema.

Como características principales de los algoritmos, puede decirse qué:

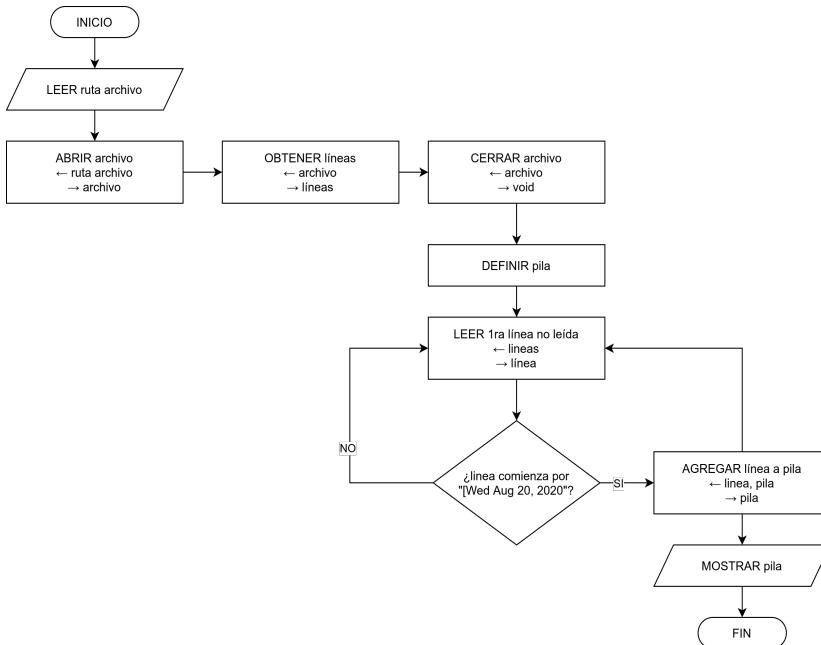
1. comienzan aceptando los datos de entrada que serán procesados;
2. las instrucciones definidas deben ser completas, precisas y concretas;
3. el tiempo total empleado por un algoritmo para llevar a cabo sus operaciones debe ser finito, así como la cantidad de instrucciones y el número de repeticiones cuando las hubiera;
4. finalmente, debe producir al menos un resultado (valor de salida).

Un factor clave de la programación es diseñar los algoritmos con una completa abstracción del lenguaje. Por ejemplo, si tras el análisis de un problema se determina que es necesario visualizar una pila con las entradas del registro de errores del archivo `errors.log` del servidor HTTP de Apache del día jueves 20 de agosto de 2020, considerando que cada

entrada del registro necesaria es una línea que comienza por [Thu Aug 20, 2020, se podría resolver, independientemente del lenguaje a utilizar, con un algoritmo como el siguiente:

1. LEER ruta del archivo
2. ABRIR archivo
3. OBTENER líneas del archivo leído
4. CERRAR archivo
5. DEFINIR pila
6. SI la primera línea no leída comienza por "[Wed Aug 20, 2020"
AGREGAR línea a pila
7. Repetir el paso 6 por cada línea no leída hasta que no queden líneas sin leer
8. MOSTRAR la pila en pantalla

Para facilitar la comprensión de un algoritmo, puede representarse en un *diagrama de flujo* como el siguiente:



Estos diagramas emplean diversas figuras tales como:

- *paralelogramos*, para los valores de entrada y salida;
- *rectángulos*, para las operaciones;
- *rombos*, para los condicionales;
- y *rectángulos con bordes redondeados*, para indicar inicio y fin del algoritmo.

Posteriormente, los algoritmos son traducidos a código fuente, y aquellas operaciones no disponibles de forma nativa en el lenguaje, requerirán el diseño de un nuevo algoritmo.

El siguiente código es la traducción correspondiente al diagrama de flujo anterior:

```
def obtener_registros(ruta_archivo):  
    archivo = open(ruta_archivo, "r")  
    lineas = archivo.readlines()  
    archivo.close()  
    pila = []  
    for linea in lineas:  
        if linea.startswith("[Wed Aug 20, 2020]"):  
            pila.append(linea)  
  
    print(pila)
```

El *código fuente* es el conjunto de órdenes escritas en un determinado lenguaje de programación, que componen un programa. Los *lenguajes de programación* son lenguajes informáticos diseñados para que un ordenador pueda llevar a cabo operaciones concretas. Los *lenguajes informáticos* son lenguajes artificiales creados exclusivamente para ser entendidos por los ordenadores. Los de programación son los que permiten la ejecución de órdenes pero los hay de otros tipos. Se manejan en dos niveles básicos: *bajo nivel*, que se refiere al código

máquina (código binario) y los de *alto nivel* que poseen una estructura humanamente legible e independiente del hardware.

Los hay *interpretados* (sus instrucciones son ejecutadas directamente sobre el sistema operativo por un intérprete) y *compilados* (aquellos que requieren ser traducidos a un lenguaje intermedio y empaquetados para su ejecución). Algunos funcionan en diversas plataformas y otros son exclusivos de sistemas concretos. Algunos aceptan solo un paradigma (técnica) de programación, mientras que otros, pueden aceptar más de uno.

Si bien al momento de traducir un algoritmo a código fuente la sintaxis empleada dependerá exclusivamente del lenguaje de programación elegido, todo lenguaje posee una serie de elementos comunes —a todos los lenguajes— que se encuentran disponibles. Solo la sintaxis del lenguaje es lo que cambia. Por ejemplo, mientras que una variable `foo` de tipo entero con valor 15, en Python se define mediante `foo = 15`, en PHP se define anteponiendo un signo dólar al nombre de la variable y un punto y coma tras el valor, tal que `$foo = 15;` y en C, requiere la declaración de tipo, tal que `int foo = 15;`.

El proceso de traducción de un algoritmo a código fuente en un lenguaje de programación determinado es lo que técnicamente se conoce como **codificación**. El proceso que abarca el análisis de un problema, el diseño de su solución (algoritmo) más su traducción a código fuente (codificación), es lo que se denomina **programación**. Así, el estudio de esta abarca tres grandes temas: lógica, análisis y diseño de algoritmos; y lenguajes de programación. «**Python Aplicado**» pertenece a esta tercera categoría.