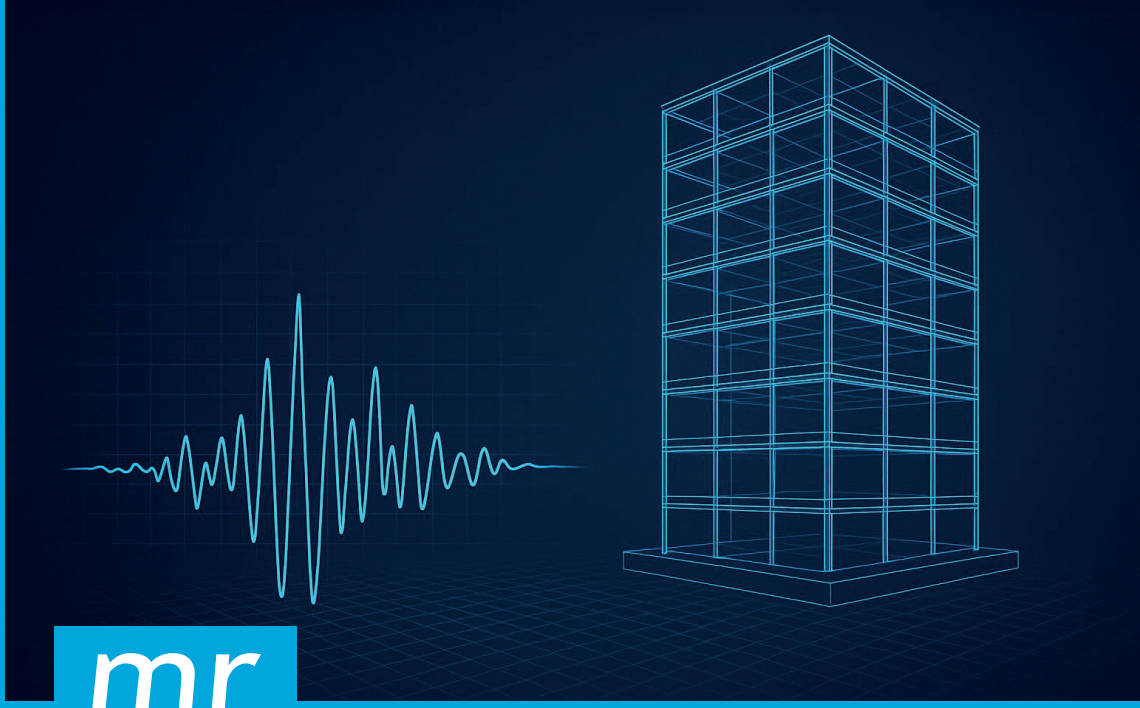

mr
manual de referencia

INGENIERÍA CIVIL, TOPOGRAFÍA Y CONSTRUCCIÓN

INTRODUCCIÓN AL CÁLCULO SÍSMICO CON OPENSEES EN PYTHON

Luis Iván Velasco Enríquez | Edgar Tapia Hernández
Antonio Hospitaler Pérez | Héctor Guerrero Bobadilla
Brian Jordano Cagua Gómez | Julia Edith Pilatasig Caizaguano
Oscar Sámano Brito | Sergio Gil Guerrero García

edUPV
Universitat Politècnica de València

INGENIERÍA CIVIL, TOPOGRAFÍA Y CONSTRUCCIÓN

Luis Iván Velasco Enríquez,
Edgar Tapia Hernández,
Antonio Hospitaler Pérez,
Héctor Guerrero Bobadilla,
Brian Jordano Cagua Gómez,
Julia Edith Pilatasig Caizaguano,
Oscar Sámano Brito,
Sergio Gil Guerrero García

Introducción al cálculo sísmico con OpenSees en Python

edUPV
Universitat Politècnica de València

http://tiny.cc/edUPV_rea

Colección Manual de referencia; serie ingeniería civil http://tiny.cc/edUPV_MR

Los contenidos de esta publicación han sido evaluados mediante el sistema doble ciego, siguiendo el procedimiento que se recoge en http://tiny.cc/Evaluacion_Obras

Para referenciar esta publicación utilice la siguiente cita:

Velasco Enríquez, Luis Iván; Tapia Hernández, Edgar; Hospitaler Pérez, Antonio; Guerrero Bobadilla, Héctor; Cagua Gómez, Brian Jordano; Pilatasig Caizaguano, Julia Edith; Sámano Brito, Oscar; Gil Guerrero García, Sergio (2025). *Introducción al cálculo sísmico con OpenSees en Python*. edUPV https://doi.org/10.4995/9788413960043edUPV

Autoría

Luís Iván Velasco Enríquez (Coordinador). *Universidad Nacional Autónoma de México*
Edgar Tapia Hernández (Revisor). *Universidad Autónoma Metropolitana*
Antonio Hospitaler Pérez (Coordinador). *Universitat Politècnica de València*
Héctor Guerrero Bobadilla (Coordinador). *Universidad Nacional Autónoma de México*
Brian Jordano Cagua Gómez. *Universidad de los Andes*
Julia Edith Pilatasig Caizaguano. *Universidad de los Andes*
Oscar Sámano Brito. *Universidad Nacional Autónoma de México*
Sergio Gil Guerrero García. *Universidad Nacional Autónoma de México*

Edición

2025, edUPV (Editorial Universitat Politècnica de València)
Ref.:6831_01_01_01

ISBN: 978-84-1396-343-3 (versión impresa)
ISBN: 978-84-1396-004-3 (versión electrónica)
DOI: https://doi.org/10.4995/9788413960043edUPV

© de los textos y las imágenes: sus autores

Si el lector detecta algún error en el libro o bien quiere contactar con los autores, puede enviar un correo a edicion@editorial.upv.es

Introducción al cálculo sísmico con OpenSees en Python / edUPV

Se permite la reutilización de los contenidos mediante la copia, distribución, exhibición y representación de la obra, así como la generación de obras derivadas siempre que se reconozca la autoría y se cite con la información bibliográfica completa. No se permite el uso comercial y las obras derivadas deberán distribuirse bajo la misma licencia que regula la obra original.

Resumen

Este libro reúne los conocimientos y experiencias de diversos investigadores e investigadoras en el uso de OpenSees, un programa de análisis estructural ampliamente reconocido en la comunidad académica internacional por su eficiencia para realizar simulaciones sísmicas. Comienza con una introducción a Python, donde se presentan los comandos y estructuras básicas del lenguaje y se desarrolla a través de ejemplos que abordan temas como análisis estáticos y dinámicos, efectos no lineales (tanto geométricos como de materiales), evaluación del desempeño sísmico de estructuras e implementación de sistemas de protección sísmica. Cada uno de estos ejemplos incluye tanto una introducción teórica de los conceptos tratados como el código necesario para su reproducción.

Considerando las tendencias actuales en programación, todos los ejemplos expuestos se realizan mediante la librería OpenSeesPy, que permite ejecutar OpenSees en un entorno Python.

Gracias a su enfoque de aproximación gradual a OpenSees y a la incorporación de conceptos de dificultad creciente, esta obra está pensada para ser de utilidad tanto a estudiantes como a docentes e investigadores que deseen profundizar en el campo de la ingeniería sísmica.

A todas aquellas personas que nos han inspirado,
aconsejado y apoyado para su realización.
A nuestras familias, amigos y mentores, gracias
por su paciencia y por creer en nosotros.
A nuestros lectores, ya sean docentes
o estudiantes de ingeniería sísmica,
les deseamos que encuentren en estas páginas
el conocimiento y la experiencia
que buscamos compartir.

Agradecimientos

Los autores agradecen el apoyo de la Universitat Politècnica de València por brindarnos las facilidades y los recursos necesarios para la publicación de este libro.

Brian Cagua agradece el apoyo de ANID-Subdirección de Capital Humano/ Doctorado/2022-folio 21220089, por la beca para realizar sus estudios de posgrado en Chile.

Julia Pilatasig agradece el apoyo de ANID-Subdirección de Capital Humano/ Doctorado/2024-folio 21240148, por la beca para realizar sus estudios de posgrado en Chile.

Luis Velasco agradece el apoyo de la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI) (CVU: 897464), por la beca para realizar sus estudios de posgrado en México.

Prefacio

América Latina es una región que se ha caracterizado por fomentar históricamente modelos educativos basados en instituciones públicas y gratuitas. Gracias a este compromiso con la accesibilidad del conocimiento, hoy en día la región no solo cuenta con instituciones públicas de educación superior reconocidas a nivel mundial, sino que también ha gestado publicaciones científicas que cumplen con las características del modelo *OpenAccess* desde antes de que el término fuera acuñado. Por lo anterior, los latinoamericanos debemos sentir un profundo orgullo, sin embargo, a la vez debemos ser conscientes de que la gratuidad del conocimiento es un ideal que nos exige renovar nuestro compromiso de manera constante.

Siguiendo los principios que nos han inculcado nuestras alma mater, el presente libro es un texto docente orientado para la formación de ingenieros estructurales, completamente gratuito y accesible para todos aquellos que se encuentren interesados en aprender a utilizar el programa de elementos finitos OpenSees en temas relacionados con la ingeniería sísmica. Para abarcar la amplia variedad de temas que pueden abordarse con OpenSees, este libro es el resultado de una colaboración entre académicos y académicas de la Universitat Politècnica de València, la Universidad Nacional Autónoma de México, la Universidad Autónoma Metropolitana y la Universidad de los Andes. Gracias a su conocimiento y experiencia, el libro aborda una diversidad de temas de ingeniería estructural que van desde análisis lineales de estructuras sencillas, hasta conceptos más complejos como la realización de análisis no lineales o la implementación de sistemas de protección sísmica.

En cada ejemplo se brinda una descripción de la estructura, del código que construye al modelo y una interpretación de los resultados. Adicionalmente, cada capítulo está enriquecido con un trasfondo teórico, lo que garantiza una completa comprensión de los resultados obtenidos por parte de las y los lectores. Para permitir su replicabilidad, estos pueden ejecutar en su propia computadora los ejemplos presentados, descargando los archivos asociados al libro desde un repositorio público, al que se puede acceder por medio del siguiente enlace: https://github.com/LuisVelasc/Analisis-estructural-con-OpenSees-en-Python/tree/main.

Al finalizar la lectura de este libro, el lector conocerá los comandos fundamentales para el uso de OpenSees en el lenguaje de programación Python, siendo capaz de interpretar y crear sus propios modelos. También estará familiarizado con la metodología para crear y analizar modelos estructurales bajo cargas estáticas y dinámicas, tendrá nociones básicas sobre los análisis no lineales y podrá modelar y analizar estructuras considerando diversos sistemas de protección sísmica. Se espera que, a través del conocimiento de estas herramientas, las comunidades de profesionales puedan seguir desarrollando trabajos académicos y de investigación que fortalezcan la resiliencia sísmica en la región.

Esperamos que las y los ingenieros estructurales interesados en un curso de OpenSees riguroso, amplio, gratuito y en su propio idioma puedan encontrar en este libro una valiosa ayuda. Que, además, les recuerde que la difusión del conocimiento es una responsabilidad que involucra a todo aquel que lo posee.

Luis Iván Velasco Enríquez
Ciudad Universitaria, Ciudad de México
Marzo 2025

Índice

Resumen.....III

Agradecimientos VII

Prefacio IX

Parte 1. Conceptos básicos 1

1. Introducción 3

2. Breve introducción a OpenSees 7

Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*

Héctor Guerrero Bobadilla. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*

Antonio Hospitaler Pérez. *ICITECH, Universitat Politècnica de València*

2.1. Orígenes de OpenSees.....9

2.2. Ambiente de OpenSees 9

2.3. Interfaz de OpenSees.....10

2.4. Descarga de OpenSees 11

2.5. Ayudas disponibles en línea 11

3. Lenguaje de programación Python 13

Sergio Gil Guerrero García. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*

3.1. Características 15
3.2. Ventajas de Python 15
3.3. Desventajas de Python 16
3.4. Usos de Python 16
3.5. Instalación 16
3.6. Instalación de Anaconda y OpenSeesPy 17
3.7. Tipo de datos 24
3.8. Cadenas 27
3.9. Control de flujo 30
3.10. Programación funcional 32
3.11. Programación modular y paquetes 35

Parte 2. Análisis lineal 39

4. Análisis estático de una armadura 41

Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Héctor Guerrero Bobadilla. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Antonio Hospitaler Pérez. *ICITECH, Universitat Politècnica de València*

4.1. Introducción 43
4.2. Modelado de la armadura 43
4.3. Visualización del modelo 47
4.4. Registros de resultados 47
4.5. Análisis estático de la armadura 48
Anexo 1: Código del ejemplo del Capítulo 4 51

5. Análisis modal de un marco 53

Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Héctor Guerrero Bobadilla. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Antonio Hospitaler Pérez. *ICITECH, Universitat Politècnica de València*

5.1. Introducción 55
5.2. Resolución analítica 55
5.3. Modelado del marco 59
5.4. Análisis modal del marco 62
Anexo 2: Código del ejemplo del Capítulo 5 65

6. Análisis dinámico de un péndulo invertido 69

Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Héctor Guerrero Bobadilla. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Antonio Hospitaler Pérez. *ICITECH, Universitat Politècnica de València*

6.1. Modelado y análisis estático del péndulo invertido 71
6.2. Amortiguamiento de Rayleigh 73
6.3. Análisis dinámico del péndulo invertido.....75
6.4. Respuesta dinámica de la estructura.....76
6.5. Resolución analítica 78
Anexo 3: Código del ejemplo del Capítulo 6 80

7. Análisis dinámico lineal de un marco 3D 83

Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Antonio Hospitaler Pérez. *ICITECH, Universitat Politècnica de València*
Héctor Guerrero Bobadilla. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*

7.1. Modelado del marco en 3D 86
7.2. Análisis del marco en 3D 93
7.3. Respuesta dinámica del marco 94
Anexo 4: Código del ejemplo del Capítulo 7 98

Parte 3. Análisis no lineal 105

8. Diferencias entre análisis lineal y no lineal 107

Brian Jordano Cagua Gómez. *Facultad de Ingeniería y Ciencias Aplicadas, Universidad de los Andes*
Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*

8.1. Tipos de comportamientos en materiales 109
8.2. Fuentes de no linealidad.....113
8.3. Comportamiento mecánico del material 113
8.4. Consideraciones de los análisis no lineales.....118
8.5. Ejemplo sobre la respuesta axial de un material 119
Anexo 5. Código del ejemplo del Capítulo 8 124

9. Análisis de una armadura considerando efectos de no linealidad.....127

Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Antonio Hospitaler Pérez. *ICITECH, Universitat Politècnica de València*
Héctor Guerrero Bobadilla. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*

9.1. Grandes desplazamientos, pequeñas deformaciones.....129

9.2. Comportamiento no lineal de los materiales 130
9.3. Modelado y análisis de la armadura 132
9.4. Comparación de la respuesta de la armadura 135
Anexo 6: Código del ejemplo del Capítulo 9 138

10. Análisis estático no lineal de un marco de acero 143

Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Héctor Guerrero Bobadilla. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Antonio Hospitaler Pérez. *ICITECH, Universitat Politècnica de València*

10.1. Análisis estáticos no lineales 145
10.2. Modelado de estructuras con elementos finitos 147
10.3. Modelado del marco de acero 150
10.4. Análisis estático no lineal 157
10.5. Curva de capacidad 158
Anexo 7: Código del ejemplo del Capítulo 10 160

11. Diagrama momento-curvatura de una sección rectangular de concreto reforzado 165

Edgar Tapia Hernández. *Universidad Autónoma Metropolitana, Azcapotzalco*
Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*

11.1. Definición de parámetros de análisis 168
11.2. Subprocedimiento MomentCurvature 168
11.3. Definición de la sección transversal 170
11.4. Diagramas momento-curvatura 173
Anexo 8: Código del ejemplo del Capítulo 11 175

12. Análisis dinámico no lineal de un marco de concreto reforzado 179

Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Héctor Guerrero Bobadilla. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Antonio Hospitaler Pérez. *ICITECH, Universitat Politècnica de València*

12.1. Elementos con rótulas plásticas 182
12.2. Modelos de concreto confinado 186
12.3. Modelado del marco de concreto 190
12.4. Análisis dinámico no lineal 195
12.5. Resultados 196
Anexo 9: Código del ejemplo del Capítulo 12 202

Parte 4. Ingeniería sísmica 211

13. Introducción al diseño sísmico basado en desempeño y sistemas de control ...213

Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Julia Edith Pilatasig Caizaguano. *Facultad de Ingeniería y Ciencias Aplicadas, Universidad de los Andes*
Héctor Guerrero Bobadilla. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*

13.1. Diseño sísmico basado en el desempeño.....215
13.1.1. Filosofía del diseño sísmico basado en desempeño.....218
13.1.2. Evaluación del desempeño sísmico 221
13.1.3. Modelado de la estructura.....224
13.1.4. Evaluaciones de pérdidas económicas y sociales 226
13.2. Sistemas de protección sísmica 228
13.2.1. Aislamiento sísmico.....232
13.2.2. Sistemas disipadores.....233
13.2.3. Amortiguadores de masa sintonizada 236

14. Aplicación del método del espectro de capacidad 237

Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Antonio Hospitaler Pérez. *ICITECH, Universitat Politècnica de València*
Héctor Guerrero Bobadilla. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*

14.1. Método del espectro de capacidad 239
14.2. Espectros de respuesta según la NTC 244
14.3. Modelos de rótulas plásticas concentradas.....245
14.4. Modelado y análisis del marco con rótulas concentradas 246
14.5. Evaluación del desempeño del marco por el MEC 250
Anexo 10: Código del ejemplo del Capítulo 14 255

15. Análisis dinámico de un marco con contraventeo convencional 261

Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Héctor Guerrero Bobadilla. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Antonio Hospitaler Pérez. *ICITECH, Universitat Politècnica de València*

15.1. Contraventeos convencionales 263
15.2. Falla por fatiga 266
15.3. Modelado y análisis del marco con contraventeo 267
15.4. Comparación de la respuesta con y sin contraventeo.....272
15.5. Análisis con otras curvas Coffin-Manson 273
Anexo 11: Código del ejemplo del Capítulo 15 276

16. Análisis dinámico de un marco con contraventeo restringido al pandeo.....279

Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Héctor Guerrero Bobadilla. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Antonio Hospitaler Pérez. *ICITECH, Universitat Politècnica de València*

16.1. Contraventeos restringidos al pandeo 281
16.2. Modelado y análisis del marco con CRP.....284
16.3. Comparación de la respuesta con y sin contraventeo.....287
16.4. Estudio paramétrico del CRP 289
Anexo 12: Código del ejemplo del Capítulo 16 292

17. Análisis dinámico de un marco con amortiguador viscoso.....295

Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Héctor Guerrero Bobadilla. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Antonio Hospitaler Pérez. *ICITECH, Universitat Politècnica de València*

17.1. Amortiguadores viscosos 297
17.2. Modelado y análisis del marco con amortiguador viscoso 300
17.3. Comparación de la respuesta con y sin amortiguador viscoso.....302
Anexo 13: Código del ejemplo del Capítulo 17 304

18. Análisis dinámico de un marco con aislamiento sísmico 307

Oscar Sámano Brito. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*

18.1. Aisladores sísmicos.....309
18.2. Modelado y análisis del marco con aislamiento sísmico 315
18.3. Comparación de la respuesta con y sin aisladores 320
Anexo 14: Código del ejemplo del Capítulo 18 323

19. Análisis considerando efectos de interacción dinámica suelo-estructura 329

Edgar Tapia Hernández. *Universidad Autónoma Metropolitana, Azcapotzalco*
Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*

19.1. Interacción dinámica suelo-estructura.....332
19.2. Funciones de impedancia 337
19.3. Descripción del código del modelo.....341
19.4. Resultados de los análisis estáticos no lineales.....346
19.5. Resultados de los análisis dinámicos no lineales 347
Anexo 15: Código del ejemplo del Capítulo 19 348

20. Análisis dinámico incremental.....355

Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Héctor Guerrero Bobadilla. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Antonio Hospitaler Pérez. *ICITECH, Universitat Politècnica de València*

20.1. Teoría de los análisis dinámicos incrementales.....357
20.2. Aplicación del análisis dinámico incremental.....359
20.3. Resultados 360
Anexo 16.1: Código del ejemplo del Capítulo 20 363
Anexo 16.2: Código del modelo estructural para IDA.....364

Parte 5. Inteligencia artificial.....367

21. Predicción de la respuesta dinámica por redes neuronales artificiales.....369

Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Héctor Guerrero Bobadilla. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Antonio Hospitaler Pérez. *ICITECH, Universitat Politècnica de València*

21.1. Fundamentos de las redes neuronales artificiales.....371
21.2. Entrenamiento de una red neuronal artificial 373
21.3. Generación de una muestra de análisis dinámicos no lineales.....378
21.4. Descripción del código para la generación de la muestra de datos.....381
21.5. Descripción del código para el entrenamiento de la red neuronal artificial.....383
21.6. Evaluación del desempeño de la red 386
Anexo 17.1: Código del ejemplo del Capítulo 21 391
Anexo 17.2: Códigos para la creación de la red neuronal artificial.....394
Anexo 17.3: Códigos auxiliares 397

22. Optimización metaheurística de un diseño sísmico 403

Luis Iván Velasco Enríquez. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*
Antonio Hospitaler Pérez. *ICITECH, Universitat Politècnica de València*
Héctor Guerrero Bobadilla. *Instituto de Ingeniería, Universidad Nacional Autónoma de México*

22.1. Problemas de optimización 405
22.2. Estrategias evolutivas.....407
22.3. Planteamiento del problema.....409
22.4. Descripción del código.....411
22.5. Diseños sísmicos optimizados.....414
Anexo 18.1: Código del ejemplo del Capítulo 22 418

Conclusiones generales.....427

Bibliografía 429

Parte 1

Conceptos básicos

01

Introducción

Este libro se escribe con la finalidad de introducir a los nuevos usuarios de OpenSees en el modelado y el análisis de estructuras por medio de esta herramienta. Debido a que está dirigido a nuevos usuarios, el contenido tiene un enfoque introductorio y se acompaña de ejemplos de complejidad limitada. Para aclaraciones y discusiones más detalladas sobre los alcances y limitaciones de OpenSees, se recomienda consultar el manual en línea en las siguientes direcciones: https://opensees.berkeley.edu/ o https://openseespydoc.readthedocs.io/en/latest/. De hecho, algunos de los ejemplos proporcionados en este libro se basan en los presentados en el manual en línea del programa, los cuales fueron ejecutados en la versión 3.10.12 de Python y la versión 3.5.1.3 de OpenSeesPy.

El documento se divide en cinco partes: conceptos básicos, análisis lineal, análisis no lineal, ingeniería sísmica e inteligencia artificial aplicada a la ingeniería estructural. La primera parte incluye una introducción a OpenSees, junto con una guía breve sobre el uso de comandos en Python, lenguaje de programación utilizado en esta edición del libro. La sección de análisis lineal se compone de cuatro ejemplos que pretenden servir de acercamiento a los comandos de OpenSees, en su versión Python. Se realizan análisis estático, modal y dinámico lineal de cuatro estructuras.

En la parte de análisis no lineal se brindan ejemplos más complejos. Un breve ensayo sobre las diferencias entre los análisis lineales y no lineales inicia esta sección del libro y tiene la finalidad de discutir los conocimientos teóricos necesarios, la justificación y los criterios para incluir los efectos no lineales en las estructuras. Adicionalmente, la cuarta sección trata sobre ingeniería sísmica, abordándose los temas del diseño

Luis Iván Velasco Enriquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla;
Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

sísmico basado en desempeño y los sistemas de control de la respuesta. Esta sección, la más extensa del libro, incluye ejemplos relacionados con el método del espectro de capacidad, el modelado de sistemas de protección sísmica, la interacción dinámica suelo-estructura y la realización de análisis dinámicos incrementales.

Finalmente, la quinta parte del libro aborda el uso de OpenSees para el entrenamiento de redes neuronales que predigan la respuesta dinámica de estructuras y como programa de apoyo para la optimización metaheurística de diseños sísmicos. En esta sección se abordan únicamente dos ejemplos, sin embargo, el trasfondo teórico de cada uno facilitará a los lectores adentrarse por cuenta propia en estos temas tan relevantes actualmente. Un bosquejo del contenido de cada capítulo se muestra a continuación.

En los dos primeros capítulos del libro se incluye una introducción a OpenSees (Capítulo 2) y al lenguaje de programación Python (Capítulo 3). Se discute el ambiente de OpenSees, subrayando sus características y ventajas. Se explica cómo evaluar expresiones matemáticas, establecer variables y definir estructuras de control básicas en Python. Asimismo, se describe de forma detallada el proceso de instalación necesario para seguir los ejemplos presentados en el libro. Estos dos capítulos iniciales permiten al lector familiarizarse con el modelado en OpenSees y proporciona los conocimientos elementales para el manejo del lenguaje Python.

La segunda parte del libro comienza en el Capítulo 4, donde se incluye un ejemplo de modelado y análisis estático de una armadura en dos dimensiones. La estructura se modela y analiza linealmente bajo cargas estáticas. El objetivo es brindar una introducción al modelado y análisis de estructuras en OpenSees a través de un ejemplo sencillo conformado por elementos con secciones generales.

El Capítulo 5 incluye un ejemplo de determinación de los modos y frecuencias de un marco de concreto. Para ello, se modela un marco de concreto de dos niveles y una crujía. Posteriormente, se determinan las frecuencias naturales y modos de vibrar de la estructura. A través de este ejemplo, se aprende a definir los materiales y se conoce el proceso para establecer secciones elásticas en OpenSees. De manera adicional, se definen las masas y el cálculo de las frecuencias naturales de una estructura.

En el Capítulo 6 se realiza un análisis dinámico lineal de una columna, donde se modela un elemento con una sección transversal general y una masa actuante en la parte superior. Se realiza un análisis dinámico de la estructura sometiéndola a un registro de aceleraciones en su base. Este ejemplo permite conocer el procedimiento para realizar un análisis en el tiempo de una estructura sencilla, mostrando además la manera de simular movimientos sísmicos en OpenSees.

El Capítulo 7 concluye la primera parte del libro, correspondiente a ejemplos de análisis lineal, con el modelado de un marco de tres dimensiones. En este capítulo se abordan las consideraciones que deben tenerse en cuenta al analizar modelos tridimensionales y se realiza un análisis dinámico a la estructura considerando un registro de aceleraciones de dos componentes.

Luis Iván Velasco Enríquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla;
Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

El Capítulo 8 marca el inicio de la tercera parte del libro, dedicado a análisis no lineales. En él se discuten las diferencias entre el análisis lineal y no lineal. Se describen las fuentes de no linealidad y el comportamiento mecánico de los materiales. Se dan algunas recomendaciones de modelado no lineal y se desarrolla un ejemplo donde se determina la curva esfuerzo-deformación de un material sujeto a cargas cíclicas.

El Capítulo 9 se analiza la misma armadura estudiada en el Capítulo 4, pero considerando los efectos de no linealidad geométrica y de los materiales.

En el Capítulo 10 se realiza un análisis estático no lineal (también conocido como *pushover*) de un marco de tres niveles y dos crujías en 2D conformado por perfiles metálicos. A través de este ejemplo, se muestra cómo definir secciones metálicas en cajón y doble T en OpenSees. Adicionalmente, se discuten las diferencias entre los distintos elementos finitos tipo barra que nos permiten considerar efectos no lineales en OpenSees.

En el Capítulo 11 se calcula la curva momento-curvatura de una sección rectangular de concreto reforzado. Este capítulo describe el proceso de creación de secciones compuestas por diferentes materiales en OpenSees.

En el Capítulo 12 se discute el análisis dinámico no lineal de un marco de concreto reforzado. La estructura, que consta de tres niveles y tres crujías, es sometida a un movimiento sísmico en la base. El ejemplo ilustra el manejo de comandos de control para el modelado de estructuras, describe la manera de definir elementos con rótulas plásticas y aborda el concepto de daño estructural.

La cuarta parte del libro, dedicada a la ingeniería sísmica, comienza con el Capítulo 13. En ella se establecen los conceptos básicos del diseño sísmico basado en desempeño y de los sistemas de protección sísmica.

El Capítulo 14 establece un ejemplo donde se modela un marco de cinco niveles y se evalúa su desempeño por medio del método del espectro de capacidad. En este capítulo se retoman varios conceptos discutidos previamente en la tercera parte del libro y se aborda el procedimiento para evaluar el desempeño sísmico de una estructura. Asimismo, se muestra cómo definir elementos con rótulas concentradas en OpenSees.

En los capítulos 15 a 17 se presentan los análisis dinámicos de un marco equipado con diferentes sistemas de protección sísmica. El marco de concreto reforzado consta de un nivel y una crujía. En cada capítulo se realizan análisis dinámicos y se discutirán los resultados obtenidos. A través de estos ejemplos se analiza la manera de modelar contraventeos[1] convencionales, contraventeos restringidos al pandeo y disipadores viscosos.

---

[1] Los contraventeos son elementos diagonales que se utilizan para proporcionar rigidez lateral a las estructuras. En algunos países, estos elementos reciben el nombre de contravientos o elementos de arriostramiento. En particular, se señala que los contraventeos son un tipo de arriostramiento.

En el Capítulo 18 se aborda un tercer tipo de sistema de protección sísmica: el aislamiento sísmico. Para ello, se toma como ejemplo un marco plano de tres niveles y seis crujías, analizado bajo un registro de aceleraciones, considerando en primera instancia una base rígida y, posteriormente, aislada mediante el uso de aisladores de péndulo de fricción. Con esto, el ejemplo demuestra las bondades de esta clase de sistemas de protección sísmica.

En el Capítulo 19 se presenta un ejemplo de interacción dinámica suelo-estructura (IDSE), el cual es un fenómeno que ocurre cuando el suelo sobre el cual se asienta la estructura es lo suficientemente flexible como para no ser considerado rígido. Debido a la afectación que la IDSE produce en la repuesta dinámica de las estructuras, este fenómeno debe ser considerado en el análisis de estructuras apoyadas sobre suelos blandos.

En el Capítulo 20 se describe una manera de realizar un análisis dinámico incremental (ADI, o IDA por sus siglas en inglés de *Incremental Dynamic Analysis*) en OpenSees. A través de este ejemplo, se muestra una de las grandes ventajas de OpenSees: su capacidad para realizar múltiples análisis a un mismo modelo de manera automatizada.

El Capítulo 21 aborda el uso de redes neuronales artificiales para predecir la respuesta dinámica de estructuras a base de marcos. Como se discutirá en dicho capítulo, las redes neuronales son algoritmos de inteligencia artificial que requieren conjuntos de datos representativos para aprender a realizar predicciones precisas. En este contexto, es fundamental contar con herramientas eficientes de análisis estructural, como OpenSees, que permitan generar una base de datos amplia y diversa. Esta base de datos es indispensable para entrenar adecuadamente la red neuronal y asegurar su capacidad de generalización ante distintos escenarios de carga o configuración estructural.

Por su parte, el Capítulo 22 presenta un ejemplo aplicado del uso de OpenSees en un proceso de optimización metaheurística orientado al diseño sísmico. Las técnicas metaheurísticas constituyen herramientas robustas que permiten abordar problemas altamente complejos, tales como la búsqueda de diseños sísmicos que cumplan con los requisitos normativos mientras se optimiza el uso de materiales o se minimiza el impacto ambiental. Dado que estos algoritmos evalúan cientos o incluso miles de candidatos a solución a lo largo del proceso de optimización, su aplicación en ingeniería estructural requiere de códigos capaces de interactuar dinámicamente con programas de análisis estructural, permitiendo una evaluación continua y automatizada del desempeño de cada solución propuesta.

Finalmente, en el Capítulo 23 se ofrecen algunas conclusiones generales de este libro.

Luis Iván Velasco Enríquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla;
Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

02

Breve introducción a OpenSees

Luis Iván Velasco Enríquez
Instituto de Ingeniería, Universidad Nacional Autónoma de México

Héctor Guerrero Bobadilla
Instituto de Ingeniería, Universidad Nacional Autónoma de México

Antonio Hospitaler Pérez
ICITECH, Universitat Politècnica de València

Contenidos del capítulo

2.1. Orígenes de OpenSees
2.2. Ambiente de OpenSees
2.3. Interfaz de OpenSees
2.4. Descarga de OpenSees
2.5. Ayudas disponibles en línea

Luis Iván Velasco Enríquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla;
Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

2.1. Orígenes de OpenSees

En este capítulo se ofrece a los lectores una introducción general a OpenSees, con el propósito de que se familiaricen con sus componentes y funcionamiento antes de abordar los ejemplos prácticos. El programa de licencia libre OpenSees (del inglés *Open System for Earthquake Engineering Simulation*) permite analizar estructuras y modelos geotécnicos bajo cargas gravitacionales y dinámicas (PEER, 2006). Esta herramienta tiene sus orígenes en la tesis doctoral de Francis McKenna (1997). Hoy en día, gracias a las aportaciones de investigadores de diferentes universidades, así como al patrocinio y desarrollo en el Pacific Earthquake Engineering Research Center de los Estados Unidos de Norteamérica, OpenSees cuenta con un gran abanico de comandos para definir materiales, elementos y procedimientos de análisis.

A pesar de que el código de OpenSees fue definido originalmente en lenguaje C++, actualmente, los usuarios tienen la posibilidad de utilizar instrucciones en TCL (del inglés *Tool Command Language*) o Python para hacer uso del programa. Con este panorama, un conocimiento básico en programación es deseable para obtener un adecuado rendimiento de OpenSees. Al ser un programa de licencia libre, este se encuentra en continuo desarrollo y algunas de las estructuras de sus comandos pueden variar con el paso del tiempo. Sin embargo, en la página web del programa es posible consultar un manual con información actualizada con las modificaciones que se han realizado en el mismo (OpenSees, 2021). Debido a lo anterior, se recomienda recurrir a dicha página oficial de manera frecuente.

2.2. Ambiente de OpenSees

Para modelar y analizar un sistema estructural, el usuario debe definir tres conjuntos diferentes de elementos en OpenSees. Estos son *ModelBuilder*, *Analysis* y *Recorder*, comunicados por el objeto *Domain* (Velasco y Guerrero, 2020). La Figura 2.1 muestra un esquema simplificado de estos elementos.

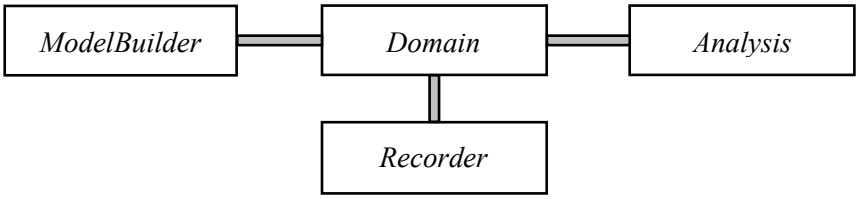

Figura 2.1. *Ambiente de OpenSees*
Fuente: elaboración propia

- ***ModelBuilder.*** Es la parte del programa donde se configuran los elementos para construir el modelo; es decir, define nodos, restricciones, masas, materiales, secciones, elementos, patrones de carga, etc. Una vez creado, el modelo se almacena en el objeto *Domain* y éste, a su vez, lo provee a los objetos *Analysis* y *Recorder*.

Luis Iván Velasco Enriquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla;
Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

- ***Analysis***. Determina el estado del modelo para cada instante de tiempo, t. Este se compone por diferentes objetos que definen el tipo de análisis que realizará el programa.
- ***Recorder.*** Este objeto crea y guarda el registro de los diferentes estados del modelo durante su análisis.

2.3. Interfaz de OpenSees

En la Figura 2.2 se muestra la interfaz del programa. Como ya se mencionó, OpenSees utiliza líneas de código para crear y analizar modelos. Esta característica permite que el programa se apoye principalmente en editores de texto. Anteriormente, era posible descargar un entorno de desarrollo llamado CypressEditor (Shojaie, 2018), que facilitaba la tarea de crear modelos para OpenSees en lenguaje TCL. Desafortunadamente, la página oficial de este ambiente de desarrollo se encuentra en desuso, por lo que la obtención de su instalador se ha vuelto complicada. Otra alternativa para la creación de modelos en OpenSees es por medio de Python, un lenguaje de programación mucho más popular que TCL. Como ambiente de desarrollo para Python se recomienda utilizar Spyder (2021), u otras alternativas como JupyterNotebook (Project Jupyter, 2022).

Sin importar el lenguaje de programación o ambiente de desarrollo utilizado, OpenSees no incluye una interfaz gráfica integrada que permita la visualización de los modelos creados. Sin embargo, tanto en lenguaje TCL como en Python, es posible utilizar un conjunto de líneas de comando para generar una representación gráfica del modelo construido. Adicionalmente, existen programas adicionales que añaden sus propias interfaces gráficas a OpenSees, como OpenSees Navigator (Yang y Schellenberg, 2015) o GID+OpenSees (Papanikolaou *et al.*, 2017), entre otras.

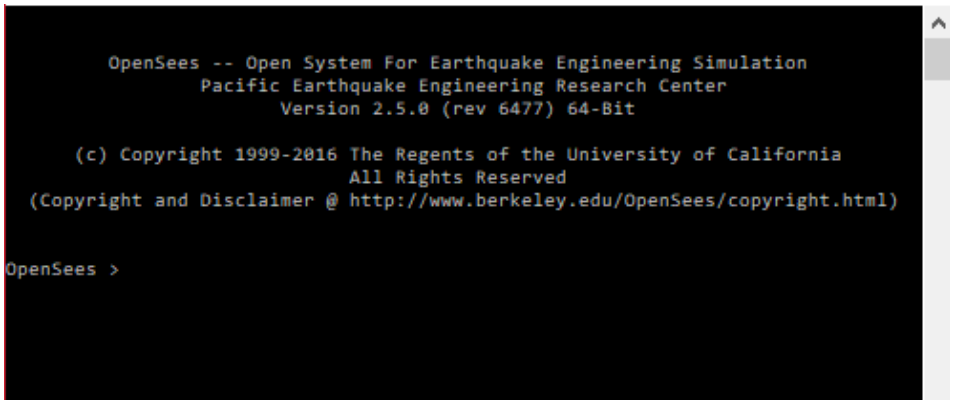

Figura 2.2. *Interfaz de OpenSees*
Fuente: elaboración propia

Luis Iván Velasco Enríquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla; Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

2.4. Descarga de OpenSees

OpenSees es un programa de licencia libre, lo que permite instalarlo y compartirlo sin ningún inconveniente. Para descargarlo es necesario acceder a su página web: http://opensees.berkeley.edu/

Una vez que el usuario se registra en el sitio, este podrá descargar dos archivos necesarios para la instalación y operación de OpenSees. El primero contiene el ejecutable del programa y el segundo las librerías de TCL. La descripción completa del proceso de descarga, junto con sus consideraciones, se encuentra en la página web.

Todos los ejemplos abordados en este libro utilizan la librería OpenSeesPy, la cual permite utilizar las funciones de OpenSees en Python (OpenSeesPy, 2019). Para tener un adecuado seguimiento de estos ejemplos, se recomienda al lector instalar el ambiente de programación Anaconda (https://www.anaconda.com/products/distribution) en sus computadoras, el cual incluye Python, y descargar la librería de OpenSees. Esto último se realiza al introducir el comando «pip install openseespy» en la consola de Python. De aquí en adelante se le denominará OpenSeesPy a la versión Python de OpenSees.

2.5. Ayudas disponibles en línea

En la página oficial de OpenSees (http://opensees.berkeley.edu/) es posible encontrar materiales de apoyo para orientar a los usuarios en el uso del programa. Las ayudas más relevantes de esta página web son:

- **Manual de inicio de OpenSees (*Getting Started Manual*).** Esta ayuda describe el proceso de descarga e instalación de OpenSees y explica los conceptos más elementales del programa.
- **Manual de los comandos de OpenSees (*Command Manual*).** En esta sección es posible encontrar la descripción de cada uno de los comandos que se emplean en OpenSees. Indica cómo usar los comandos para crear el modelo, las características de cada uno de los objetos que pueden conforman el análisis y cómo obtener los resultados.
- **Lista de ejemplos (*Examples*).** Para ayudar a los nuevos usuarios de OpenSees, en esta sección es posible encontrar ejemplos con una gran variedad de estructuras, siendo analizadas bajo diferentes tipos de solicitaciones.

Por su parte, OpenSeesPy cuenta con su propia página web, en la que se indica la estructura de los comandos empleados por la librería, junto con diferentes ejemplos: https://openseespydoc.readthedocs.io/en/latest/index.html

Luis Iván Velasco Enriquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla;
Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

03

Lenguaje de programación Python

Sergio Gil Guerrero García
Instituto de Ingeniería, Universidad Nacional Autónoma de México

Luis Iván Velasco Enríquez
Instituto de Ingeniería, Universidad Nacional Autónoma de México

Contenidos del capítulo

3.1. Características
3.2. Ventajas de Python
3.3. Desventajas de Python
3.4. Usos de Python
3.5. Instalación
3.6. Instalación de Anaconda y OpenSeesPy
3.7. Tipo de datos
3.8. Cadenas
3.9. Control de flujo
3.10. Programación funcional
3.11. Programación modular y paquetes

Luis Iván Velasco Enríquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla;
Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

Python, creado por Guido van Rossum a principios de los 90, es un lenguaje de programación de alto nivel, ya que se aproxima al lenguaje natural humano. Cuenta con una segunda versión (Python 2), que fue liberada en el año 2000. Su versión más actual, Python 3, fue liberada en 2008. Dado que los ejemplos incluidos en este libro están desarrollados en Python, resulta indispensable realizar una breve revisión de este lenguaje de programación. Por ello, en las secciones siguientes se presentarán sus conceptos fundamentales.

3.1. Características

Python posee varias características que lo diferencian de muchos otros lenguajes. Las principales son las siguientes:

- **Es interpretado.** Esto que significa que ejecuta instrucciones sin una previa compilación, mediante un intérprete.
- **Utiliza tipado dinámico.** Permite que las variables tomen valores de diferentes tipos en cualquier momento, ya que el tipo se determina en tiempo de ejecución. No es necesario declarar el tipo de dato a utilizar, a diferencia de lo que ocurre en lenguajes como C o Java.
- **Es fuertemente tipado.** Una variable que tenga un valor de un tipo concreto no se usa como si fuera de un tipo distinto al momento de aplicar operaciones sobre la variable, a menos que se haga una conversión, conocida como casteo, del inglés *casting*.
- **Es multiparadigma.** Permite adoptar varios paradigmas de programación, como son la programación estructurada, programación orientada a objetos (POO) y programación funcional.

3.2. Ventajas de Python

Las ventajas del lenguaje Python son las siguientes:

- **Fácil de usar.** Contiene todas las expresiones que se utilizan en otros lenguajes de programación, pero simplificadas.
- **Expresividad.** Una línea de código en Python puede realizar más tareas que una línea de código en muchos otros lenguajes. Esto implica mayor facilidad para mantener y depurar los programas.
- **Legibilidad.** Python utiliza una sintaxis sencilla y elegante, lo que facilita la lectura de los programas.
- **Multiplataforma.** Un mismo código puede ejecutarse tanto en Windows como en otros sistemas operativos como UNIX.
- ***Open Source.*** Se descarga e instala cualquier versión y se usa para desarrollar *software* comercial sin necesidad de tener una suscripción o el pago de licencias.

Luis Iván Velasco Enriquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla;
Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

3.3. Desventajas de Python

Por su parte, se enumeran las siguientes desventajas de Python:

- **No es el lenguaje más rápido.** Al ser un lenguaje interpretado, los programas pueden ejecutarse más lentamente en comparación con aquellos compilados en otros lenguajes. Para el desarrollo de aplicaciones que necesiten el procesamiento a altas frecuencias, existen lenguajes más eficientes.
- **No posee las librerías más extensas.** A pesar de que contiene una gran cantidad de librerías desde el momento de su instalación, otros lenguajes como C o Java tienen librerías más extensas. Sin embargo, Python puede extenderse con librerías provenientes de otros lenguajes de programación, lo que supone una gran ventaja.
- **No tiene revisión de tipos.** Las variables declaradas son referencias a objetos, por lo que no es están ligados a un tipo en particular.

3.4. Usos de Python

Entre sus principales usos están:

- Desarrollo web
- *Machine Learning*
- Visualización de datos
- Desarrollo de videojuegos
- *Scripting*
- Aplicaciones de escritorio

3.5. Instalación

Para sistemas operativos que trabajan bajo UNIX como Linux o MacOS, comúnmente ya traen una versión de Python en el sistema operativo, esto se comprueba ejecutando el siguiente comando en la terminal:

```
python –version
```

Sin embargo, es muy probable que la versión que esté instalada sea la 2.0, por lo que habría que actualizarse a la versión 3, con el siguiente comando:

Ubuntu: `sudo apt-get install python3.x.x`

Fedora: `sudo yum install python3.x.x`

Luis Iván Velasco Enríquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla; Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

En el caso de MacOS, es necesario contar con un gestor de paquetes, como podría ser *homebrew,* de esta manera, el comando es:

```
brew install python3
```

Para el caso de Windows, se descarga el instalador de Python en su versión más reciente desde su página web oficial (https://www.python.org/). Para que no haya inconvenientes con la ejecución de los programas desde la consola, hay que verificar que se elija la opción de «*add to path*» en la pantalla del instalador. Para comprobar que nuestra instalación se ha realizado con éxito, se debe abrir una consola de Windows (cmd) y teclear lo siguiente:

```
python –versión
```

Desde el punto de vista de los autores de este libro, es más recomendable instalar Anaconda (https://www.anaconda.com/products/distribution), una plataforma de desarrollo que incluye una versión de Python de manera virtual y herramientas como Spyder, ya que facilita el desarrollo de código de manera más sencilla. La manera de instalar las herramientas mencionadas se describe en profundidad en la siguiente sección.

3.6. Instalación de Anaconda y OpenSeesPy

A continuación, se explican los pasos necesarios para instalar Anaconda en Windows, configurar un entorno en Spyder e instalar las librerías requeridas para crear y analizar y visualizar modelos estructurales en OpenSees. Para poder realizar estas tareas de manera satisfactoria es importante contar con una conexión a internet. El primer paso de este proceso es descargar el instalador de Anaconda a través del siguiente enlace: https://www.anaconda.com/download.

A manera de resumen, Anaconda es una plataforma diseñada para facilitar la gestión de entornos de desarrollo. Está dirigida, principalmente, a científicos interesados en datos, ingenieros y analistas que trabajan con herramientas y bibliotecas específicas en el ámbito del análisis y visualización de datos, aprendizaje automático, inteligencia artificial, entre otros. Algunas de las características más destacadas de Anaconda son:

- **Gestión de paquetes.** Cuenta con un gestor de paquetes llamado *conda*, que permite instalar, actualizar y administrar paquetes y librerías de lenguajes como Python y R.
- **Entornos virtuales.** Permite crear y gestionar entornos virtuales aislados para Python, lo que posibilita tener diferentes configuraciones y versiones de paquetes para distintos proyectos. Esto evita conflictos entre varias versiones de paquetes y asegura que cada proyecto tenga su propio entorno limpio y controlado.

Luis Iván Velasco Enriquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla;
Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

- **Entornos de desarrollo.** Incluye dos entornos de desarrollo muy populares: Spyder (*Scientific Python Development Environment*) y Jupyter Notebook. Spyder es un Entorno de Desarrollo Integrado especialmente diseñado para el análisis numérico y científico en Python, mientras que Jupyter Notebook es una herramienta interactiva que permite crear documentos interactivos que combinan código ejecutable, texto y visualizaciones en un único entorno.

El proceso de instalación de Anaconda es bastante sencillo y no representa un reto extraordinario. Una vez descargado el instalador, basta con ejecutarlo y seleccionar las opciones deseadas. En la Figura 3.1, a manera de referencia, se muestran capturas de pantalla del inicio y del final del proceso de instalación.

Figura 3.1. *Instalación de Anaconda*
Fuente: elaboración propia

Al instalarse Anaconda, el usuario podrá hacer uso de los entornos de desarrollo Spyder y Jupyter Notebook, además del resto de herramientas que vienen incluidas. Para hacer uso de estos programas hay que abrir el navegador de Anaconda y ejecutar el entorno deseado. Para los ejemplos de este libro se hará uso de Spyder, el cual se encuentra en el navegador de Anaconda, como se muestra en la Figura 3.2.

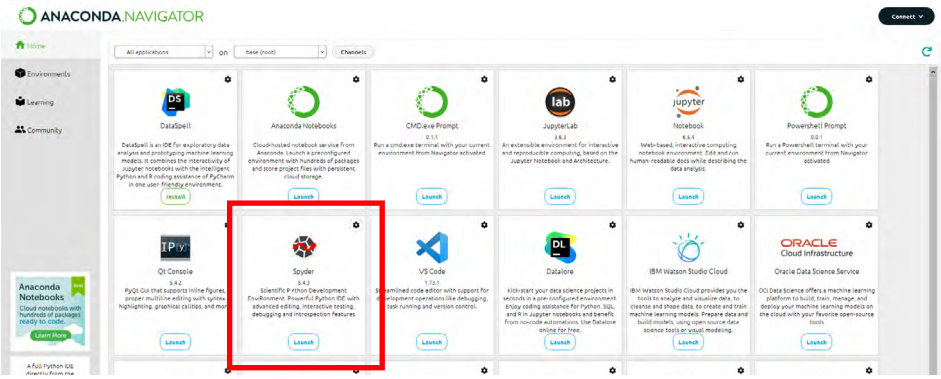

Figura 3.2. *Navegador de Anaconda*
Fuente: elaboración propia

Luis Iván Velasco Enríquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla; Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

Al abrir Spyder, el usuario podrá encontrarse con la pantalla mostrada en la Figura 3.3. Como se observa, Spyder es un entorno de desarrollo que cuenta con una estructura semejante a la de Matlab. En Spyder es posible encontrar un espacio de trabajo donde desarrollar código, un explorador de variables, figuras y archivos, además de una consola. Un punto importante a destacar es que Spyder cuenta con una versión de Python preinstalada, la misma que se identifica en la consola, en el área señalada por el recuadro rojo de la Figura 3.3. Para hacer uso de otras versiones de Python, es necesario recurrir a los entornos o ambientes.

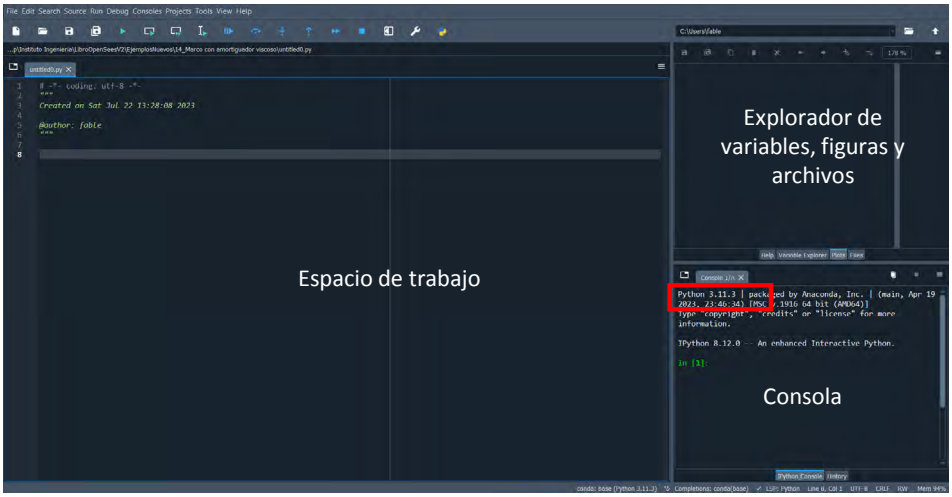

Figura 3.3. *Interfaz de Spyder*
Fuente: elaboración propia

Los ambientes en Anaconda (también conocidos como entornos) son espacios aislados que permiten organizar y gestionar proyectos de manera eficiente. Cada ambiente es una entidad aislada que contiene su propia versión de Python y de los paquetes a utilizar, lo que evita conflictos al trabajar en proyectos con diferentes configuraciones de paquetes. Para explicar de manera más detallada la idea, considérese que un caso donde es necesario trabajar con dos proyectos. En el primero se requiere Python en su versión 3.8 junto con versiones específicas de los paquetes pandas y Scikit-learn, mientras que en el segundo se necesita Python en su versión 3.9 y versiones diferentes de las mismas librerías. Para trabajar en ambos proyectos sin crear afectaciones entre los mismos es posible crear un ambiente para cada uno en los que se instalen las librerías requeridas con sus respectivas versiones.

Para crear un ambiente en Anaconda es necesario abrir el Anaconda Prompt. Esto se hace buscándolo desde la barra de tareas de Windows. Una vez abierto, se introduce un comando con la siguiente estructura en la que se especifica el nombre que tendrá el entorno y la versión de Python que utilizará:

conda create –name <nombre del entorno> python=<versión Python deseada>

Debido a que los ejemplos de este libro utilizan la versión 3.10 de Python, el lector deberá introducir el siguiente comando que creará un entorno denominado «py10» y que empleará la versión deseada del lenguaje. Al introducir alguno de los siguientes comandos, Anaconda Prompt solicitará al usuario confirmar la instalación de determinados paquetes, tal como se muestra en la Figura 3.4.

conda create –name py10 python==3.10

conda create –n py10 python=3.10

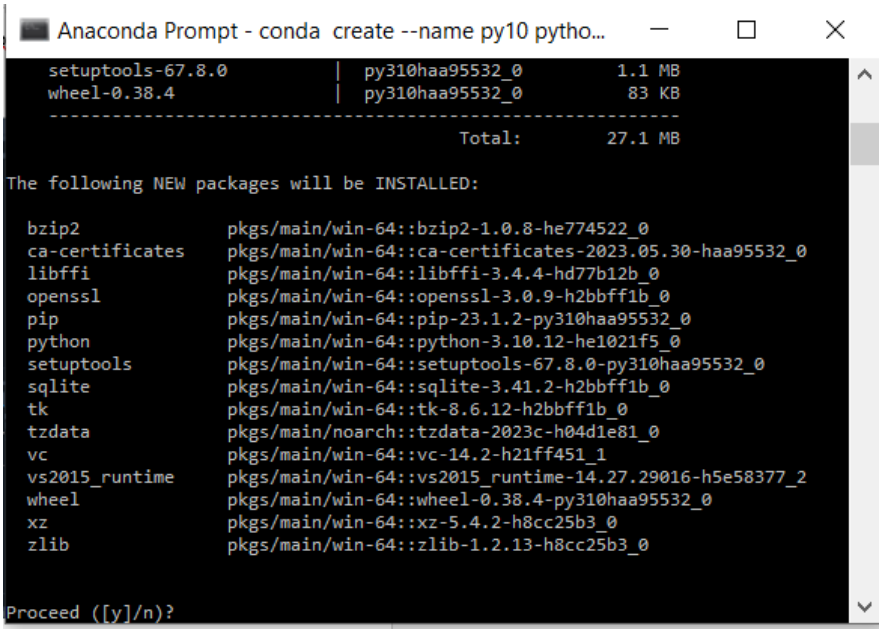

Figura 3.4. *Creación del ambiente en Anaconda Prompt*
Fuente: elaboración propia

Antes de poder hacer uso del entorno recién creado, es necesario instalar los kernels correspondientes de Spyder. La versión del kernel a descargar depende de la versión de Spyder utilizada. El número de la versión se puede encontrar en el navegador de Anaconda, en el lugar señalado por el recuadro rojo de la Figura 3.5. Una vez conocida la versión de Spyder, la versión del kernel a descargar se localiza en el siguiente enlace: https://docs.spyder-ide.org/current/troubleshooting/common-illnesses.html.

Luis Iván Velasco Enríquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla; Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

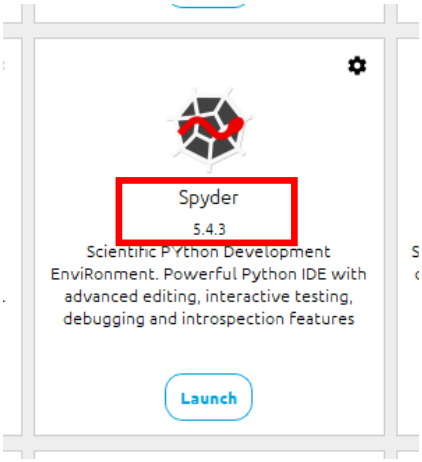

Figura 3.5. *Versión de Spyder*
Fuente: elaboración propia

La descarga e instalación del kernel de Spyder se realiza desde Anaconda Prompt. Para realizar este paso primero es necesario activar el entorno donde se va a instalar. Esto se hace introduciendo en Anaconda Prompt el siguiente comando:

conda activate <nombre del entorno>

En nuestro caso, para activar el entrono «py10», sería:

conda activate py10

Posteriormente, se descarga e instala el kernel correspondiente con el comando:

conda install spyder-kernels=<versión correspondiente>

Para el ejemplo presentado, se requiere la versión del kernel 2.4.3 por utilizarse Spyder en su versión 5.4.3, por lo tanto, se introduce el comando:

conda install spyder-kernels=2.4.3

Al introducir el comando, el *prompt* solicitará confirmar el proceso de instalación. Al finalizar es necesario reiniciar Spyder (en caso de tenerlo abierto) y activar el ambiente creado. Para cambiar el ambiente se debe seguir la siguiente ruta desde Spyder: Tools → Preferences → Python interpreter. Al seleccionar Python interpreter aparecerá la ventana mostrada en la Figura 3.6. En dicha ventana se debe seleccionar la opción «use the following Python interpreter», y seleccionar el ambiente que se ha creado. Un ejemplo de este paso se muestra en la Figura 3.7.

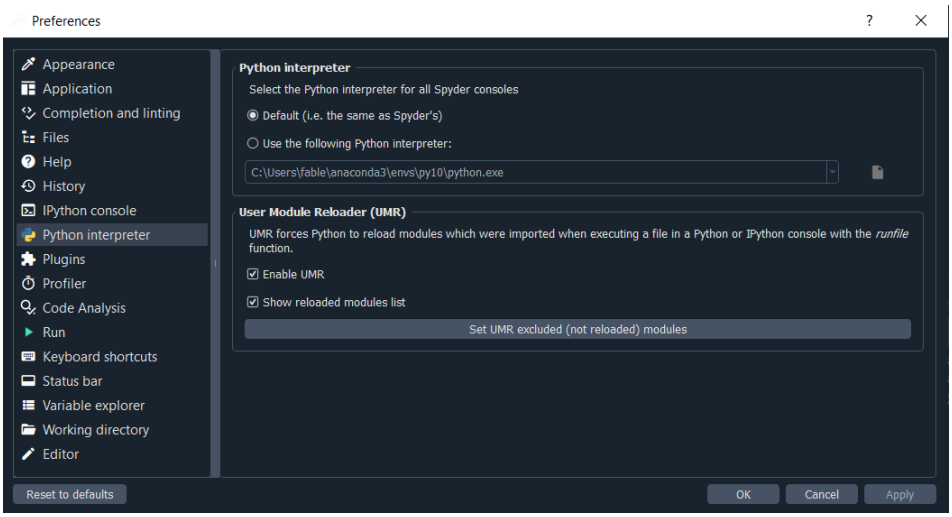

Figura 3.6. *Intérprete de Python*
Fuente: elaboración propia

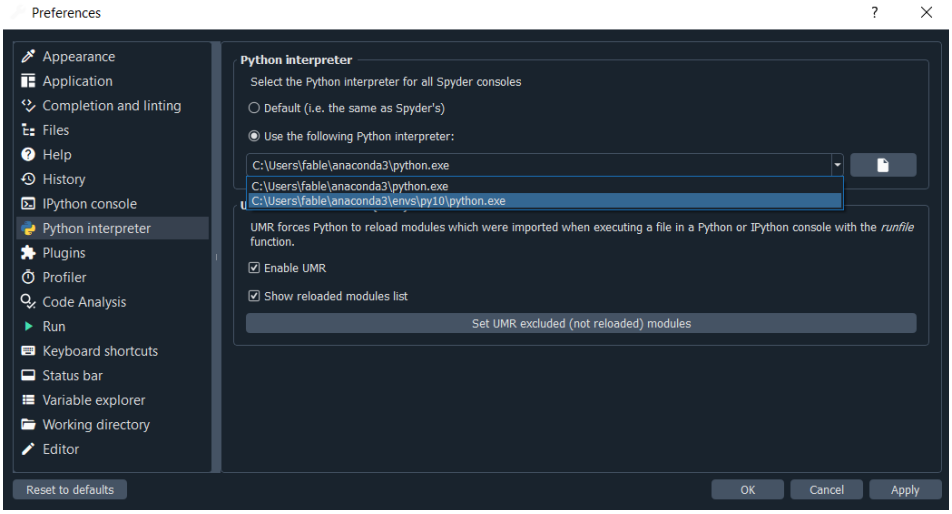

Figura 3.7. *Cambio de ambiente en Spyde*
Fuente: elaboración propia

Posteriormente, desde la consola se debe de reiniciar el kernel, tal como se muestra en la Figura 3.8. Si los pasos antes descritos fueron realizados de manera adecuada, en la nueva consola de Spyder se debería de indicar que se está utilizando una versión 3.10 de Python. Con esto, es posible comenzar a crear código con la versión 3.10 Python desde Spyder.

Luis Iván Velasco Enríquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla;
Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

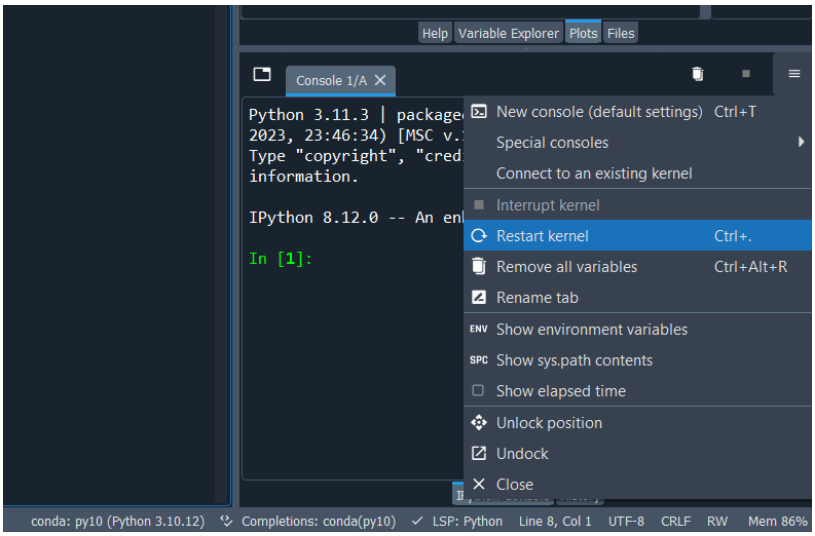

Figura 3.8. *Reinicio del kernel en Spyder*
Fuente: elaboración propia

A manera de ejemplo, en la Figura 3.9 se muestra el error que aparecería en caso de que el kernel de Spyder no se hubiera instalado correctamente o en caso de que se hubiera omitido dicho paso. Como se puede observar, sin el kernel correspondiente no es posible ejecutar código de Python en Spyder.

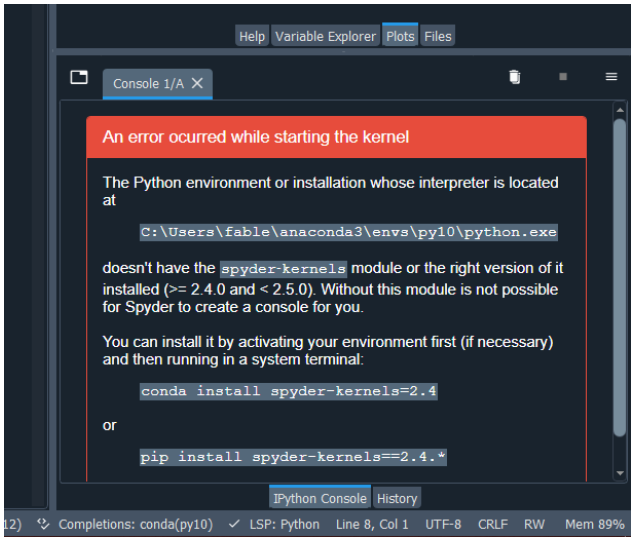

Figura 3.9. *Error por falta de los kernels en Spyder*
Fuente: elaboración propia

El paso final es instalar las librerías de Openseespy y vfo (vizualisation for opensees), las cuales nos permitirán crear modelos estructurales y visualizarnos, respectivamente. Estas instalaciones se realizan con los siguientes comandos desde la consola de Spyder:

pip install openseespy
pip install vfo

Para verificar las versiones de las librerías instaladas en Spyder se utilizar el comando pip list desde la consola. Al introducir el comando aparecerá una lista con todas las librerías instaladas con sus respectivas versiones. Como se mencionó anteriormente, en los ejemplos de este libro se utiliza la librería OpenSeesPy en su versión 3.5.1.3. Esta versión es la misma para la librería OpenSeesPyWin que se instala en conjunto de manera automática con OpenSeesPy. En caso de que se cuente con otra versión para alguna de las dos librerías, la instalación de la versión mencionada se puede forzar utilizando el comando:

pip install openseespy==3.5.1.3

3.7. Tipo de datos

Existen dos tipos de datos principales en Python, los mutables y los inmutables. Este concepto se utiliza mucho en Python, ya que explica que una clase es *inmutable* si cada objeto de esa clase tiene un valor fijo después de su instanciación; es decir, no podrá ser cambiado. En caso contrario, el concepto de *mutabilidad* nos indica que podremos cambiar el valor del objeto después de ser instanciado, o bien, asignado a una variable.

Constante None

None es una constante especial que se encuentra en el lenguaje Python, cuyo valor es nulo. Este tiene su propio tipo de dato (NoneType). A las variables podemos asignarles None y al ser un valor nulo, todas las variables que contengan a None, son iguales entre sí. Debido a varias condiciones se debe especificar lo siguiente:

- None no es lo mismo que False.
- None no es 0.
- None no es una cadena vacía.
- Si se compara a None con otra cosa que no tenga un valor None, siempre se obtendrá False.
- None es el único valor nulo.

Luis Iván Velasco Enríquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla; Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

Datos numéricos

Python posee tres tipos principales de datos numéricos, los cuales son enteros, flotantes y complejos. Con ellos se realizan operaciones con los operadores aritméticos:

- Suma (+)
- Resta (-)
- Multiplicación (*)
- División (/)
- Módulo (%), el cual es el residuo de la división
- Exponenciación (**)

A continuación, se muestran ejemplos sencillos de diferentes operaciones aritméticas, donde el símbolo → indica el resultado de la operación.

```
x = 3
y = 8
print(x + y)     → 11
print(x - y)     → -5
print(x * y)     → 24
print(x / y)     → 0.375
print(x % y)     → 3
print(x ** y)    → 6561
```

También es posible realizar operaciones con números complejos:

```
a = 5 + 6j
b = 7 + 2j
print(a + b)     → (12+8j)
```

Datos booleanos

Un dato booleano tiene solo dos posibles valores: *verdadero* o *falso*. En Python se tienen las palabras reservadas True (verdadero) y False (falso) para representar dichos estados. Este tipo de datos es utilizado principalmente para los condicionales, los cuales se explican más adelante en este documento. Se asigna un valor booleano a una variable de la siguiente manera:

```
x = True
y = False
```

Existen operaciones que se pueden realizar con los booleanos para evaluar expresiones y así determinar si es falso o verdadero, estas son *and*, *or* y *not*.

- and: 'y' lógico. Este operador da como resultado True si, y sólo si, sus dos operandos son True.

Luis Iván Velasco Enriquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla;
Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

```
print(True and True)     →    True
print(True and False)    →    False
print(False and True)    →    False
print(False and False)   →    False
```

- or: 'o' lógico. Este operador da como resultado True si algún operando es True.

```
print(True or True)      →    True
print(True or False)     →    True
print(False or True)     →    True
print(False or False)    →    False
```

- not: negación. Este operador da como resultado True si, y solo si, su argumento es False

```
print(not True)          →    False
print(not False)         →    True
```

Operadores relacionales

Para realizar comparaciones entre datos numéricos se utilizan los operadores relacionales como en matemáticas, los cuales se muestran en la Tabla 1.1.

Tabla 3.1. *Símbolos de operadores relacionales*
Fuente: elaboración propia

Relación	Símbolo
Mayor que	>
Mayor o igual que	>=
Menor que	<
Menor o igual que	<=
Equivalente a	==
Diferente de	!=

```
a = 5
b = 10
print(a >  b  )          →    False
print(a >= b  )          →    False
print(a <  b  )          →    True
print(a <= b  )          →    True
print(a == b  )          →    False
print(a != b  )          →    True
```


Luis Iván Velasco Enríquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla; Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

3.8. Cadenas

También es posible asignarle una cadena de texto a una variable, comúnmente conocidas como *strings*. Existen varias maneras para realizar la asignación, ya que podemos hacerlo tanto con comillas dobles como con comillas simples.

```
cadena = "Me gusta Python"
print(cadena)            →    Me gusta Python
```

Es posible concatenar cadenas de la siguiente manera:

```
c1 = "Mi nombre es: "
c2 = "Robert"
print(c1 + c2)           →    Mi nombre es: Robert
```

Al utilizar el operador « * », lo que sucede es que se concatena la misma cadena el número de veces que se especifica.

```
print(c2 * 4  )          →    RobertRobertRobertRobert
```

Es importante señalar que para delimitar una cadena siempre será entre comillas dobles (Por ejemplo: *"cadena"*) o con comillas simples (Por ejemplo: *'cadena'*). No hay diferencia entre si se usa una forma u otra.

Conversión de cadenas a números

Debido a la necesidad de trabajar con números y cadenas, existen métodos para la conversión de tipos, en este caso, de cadenas a números. Si se propone una cadena que representa un número (por ejemplo: `cadena = "123"`), esta no podrá ser utilizada para operaciones numéricas, ya que es una cadena. Para poder convertirla a un número, se tienen los siguientes métodos:

- int(). Convierte una cadena a un entero.
- float(). Convierte la cadena a un flotante.

```
numero_entero = int(cadena)
print(numero_entero)     →    123
numero_flotante = float(cadena)
print(numero_flotante)   →    123.0
```

Métodos asociados a cadenas

Para el uso de cadenas contamos con diversos tipos de métodos, como los son split() o joint().

Luis Iván Velasco Enríquez; Edgar Tapia Hernández; Antonio Hospitaler Pérez; Héctor Guerrero Bobadilla;
Brian Jordano Cagua Gómez; Julia Edith Pilatasig Caizaguano; Oscar Sámano Brito; Sergio Gil Guerrero García / edUPV

Para seguir leyendo, inicie el proceso de compra, click aquí