

Clase 1: Programación en Python

Curso de Python para la Ciencia de Datos
4ta Jornadas de Ingeniería Estadística UBB

Rodrigo Salas Fuentes, Dr. -Ing.

¹Escuela de Ingeniería C. Biomédica. Universidad de Valparaíso

²Centro de Investigación y Desarrollo en Ingeniería en Salud **CINGS-UV**

³Instituto Milenio Intelligent Healthcare Engineering - **iHealth**

rodrigo.salas@uv.cl



Tabla de Contenidos

- 1 Contenido del Curso
- 2 Profesor
- 3 Material de Estudio
- 4 Breve Historia de Python
- 5 Instalación de Python
- 6 Toolboxes
- 7 Lenguaje de Programación
 - Operaciones Matemáticas
 - Operaciones Relacionales
 - Sentencias de Fluxos de Control
 - Funciones
- 8 Algoritmos
- 9 Taller

Contenido del Curso

Contenido	Fechas
<p>Introducción a Python</p> <p>Historia de Python</p> <p>Instalación de Python: Anaconda, Google Colab.</p> <p>VARIABLES, EXPRESIONES, SENTENCIAS EN PYTHON</p> <p>ESTRUCTURAS DE CONTROL</p> <p>ESTRUCTURAS DE DATOS</p> <p>FUNCIONES</p> <p>Taller: "Programación con Python"</p>	19 de enero, 2022
<p>ANÁLISIS EXPLORATORIO DE DATOS CON PYTHON</p> <p>TOOLBOX DE CIENCIA DE DATOS: PANDAS, SCIKIT-LEARN, SCIPY, NUMPY, STATMODELS</p> <p>MANIPULACIÓN DE DATOS CON PYTHON</p> <p>ANÁLISIS EXPLORATORIO DE LOS DATOS</p> <p>TÉCNICAS DE CLUSTERING</p> <p>Taller: "Análisis Exploratorio de datos con Python"</p>	20 de enero, 2022
<p>ANÁLISIS MULTI-DIMENSIONAL DE DATOS CON PYTHON</p> <p>ANÁLISIS DE COMPONENTES PRINCIPALES</p> <p>TÉCNICAS DE PROYECCIÓN DE DATOS</p> <p>Taller: "Análisis Multi-Dimensional de Datos"</p>	21 de enero, 2022

Profesor: Rodrigo Salas Fuentes

- Ingeniero C. Informático. UTFSM (2002)
- Doctor en Ingeniería Informática. UTFSM (2010)
- Profesor Titular, Escuela de Ingeniería C. Biomédica, Universidad de Valparaíso. (since 2007)
- Investigador Principal del Centro de Investigación y Desarrollo en Ingeniería en Salud (CINGS-UV)
- Investigador Principal del Instituto Milenio Intelligent Healthcare Engineering (iHealth).
- Áreas de Investigación: Ciencias de la Computación, Inteligencia Artificial, Ciencia de Datos.



- e-mail: rodrigo.salas@uv.cl
- Google Scholar
- LinkedIn: [rodrigo-salas-fuentes](https://www.linkedin.com/in/rodrigo-salas-fuentes/)
- Twitter: [@rod_salas_f](https://twitter.com/rod_salas_f)
- GitHub: [link](https://github.com/rodrigo-salas-fuentes)
- Web-Page: [link](https://rodrigo-salas-fuentes.github.io)

Material de Estudio I

Apuntes del Curso:

El material del curso se encuentra disponible en mi Github personal:

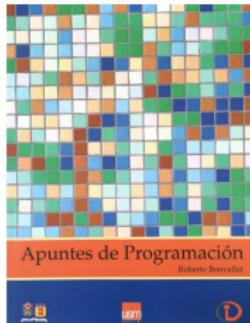


<https://github.com/rodsalasf/Python>

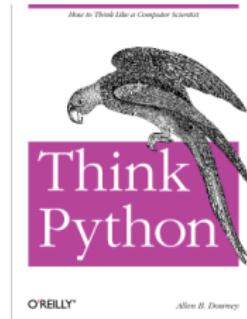
Material de Estudio II

Libros de Programación:

- Bonvallet, Roberto (2013). *Apuntes de programación*. Editorial USM.
ISBN 978-956-7051-67-0. [\[link\]](#)



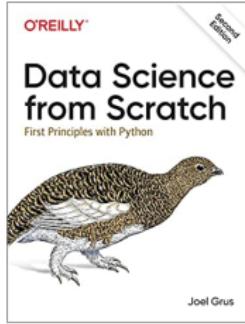
- Downey, Allen; Elkner, Jeffrey; Meyers, Chris. *Aprenda a pensar como un programador con Python*. Green Tea Press, Wellesley, Massachusetts. [\[link\]](#)
- Downey, Allen. *Think Python. How to think as a Computer Scientist*. Green Tea Press, Wellesley, Massachusetts. [\[link\]](#)



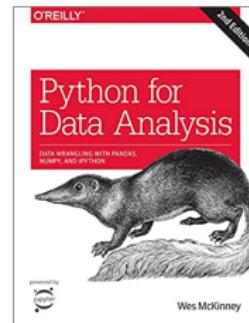
Material de Estudio III

Libros de Programación para la Ciencia de Datos

- Grus, Joel (2017). *Data Science from Scratch: First Principles with Python*. 2nd Edición. O'Reilly.



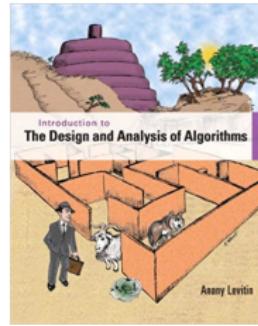
- McKinney, Wes (). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. 2nd Edición. O'Reilly.



Material de Estudio IV

Libros de Algoritmos:

- Levitin, Anany . Introducción al Análisis y Diseño de Algoritmos. 3era edición. Pearson



Breve Historia de Python

Python

Python es un lenguaje de programación interpretado de propósito general de alto nivel.



- Su filosofía de diseño enfatiza la **legibilidad del código** con el uso de una identación significativa.
- Los constructos del lenguaje, así como su enfoque **orientado a objetos**, tienen como objetivo ayudar a los programadores a escribir código claro y lógico para proyectos de pequeña y gran escala.

Historia de Python

- Creado por Guido van Rossum
- Liberado por primera vez en el año 1991 como Python 0.9.0.
- comp.lang.python se fundó en el año 1994
- Python 2.0 fue liberado en el año 2000.
- Python 3.0 fue liberado en el año 2008.
- Python 2 fue descontinuado con la versión 2.7.18 en el año 2020.
- Código abierto desde sus inicios.



Ranking de Popularidad de Lenguajes de Programación

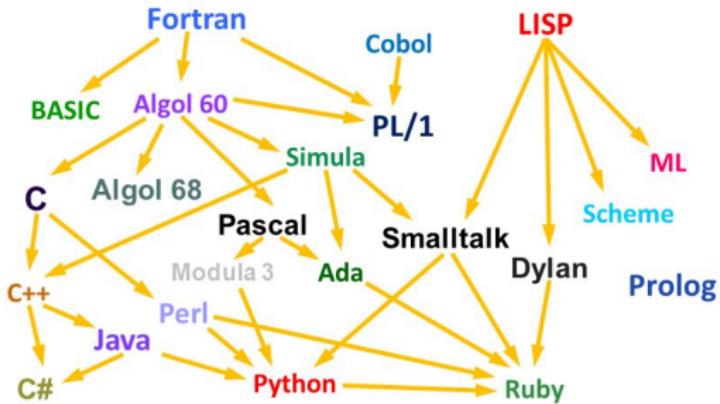
Ranking 2021 IEEE Spectrum de la popularidad de los Lenguajes de Programación.



Árbol genealógico de los lenguajes de programación

- Python fue concebido como el sucesor del lenguaje de programación ABC.
- Python ha sido influenciado por: ABC, Ada, ALGOL 68, APL, C, C++, CLU, Dylan, Haskell, Icon, Java, Lisp, Modula-3, Perl, Standard ML

A family tree of languages



Características del Lenguaje

- Lenguaje de Scripting
- Orientado a Objetos
- Interpretado
- Interactivo
- Dinámico
- Funcional
- Altamente legible
- Escalable
- Altamente extensible

Software I

- ① **Python 3.X** Python 3.X usando el framework de ANACONDA
<https://www.anaconda.com/download/> o el framework de MINICONDA
<https://docs.conda.io/en/latest/miniconda.html>
- ② **Numpy:** Computación Numérica <https://numpy.org>
- ③ **Scipy:** Computación Científica <https://www.scipy.org>
- ④ **Matplotlib:** Biblioteca completa para crear visualizaciones estáticas, animadas e interactivas en Python. <https://matplotlib.org>
- ⑤ **Seaborn:** Biblioteca de visualización de datos basada en matplotlib. Proporciona una interfaz de alto nivel para realizar gráficos estadísticos atractivos e informativos. <https://seaborn.pydata.org>
- ⑥ **Pandas:** Herramienta para análisis y manipulación de datos.
<https://pandas.pydata.org>
- ⑦ **Statmodels:** Herramientas para la estimación de muchos modelos estadísticos diferentes, así como para realizar pruebas estadísticas y exploración de datos estadísticos. <https://www.statsmodels.org>

Software II

- ⑧ **Scikit-learn:** Toolbox para machine learning <https://scikit-learn.org/>
- ⑨ **Tensorflow:** Toolbox para deep learning <https://www.tensorflow.org>
- ⑩ **Keras:** Toolbox para deep learning <https://keras.io>

Para la programación usaremos:

- ① **Jupyter notebook** (instalado con anaconda or miniconda)
- ② **Google Colab** <https://colab.research.google.com/>

Anaconda

[Products](#) ▾[Pricing](#)[Solutions](#) ▾[Resources](#) ▾[Blog](#)[Company](#) ▾[Get Started](#)

Data science technology for human sensemaking.

A movement that brings together millions of data science practitioners,
data-driven enterprises, and the open source community.

Google Colaboratory

Te damos la bienvenida a Colaboratory

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Compartir Configuración Perfil

Índice + Código + Texto Copiar en Drive Conectar Editar

¿Qué es Colaboratory?

Colaboratory, también llamado "Colab", te permite ejecutar y programar en Python en tu navegador con las siguientes ventajas:

- No requiere configuración
- Da acceso gratuito a GPUs
- Permite compartir contenido fácilmente

Colab puede facilitar tu trabajo, ya seas **estudiante, científico de datos o investigador de IA**. No te pierdas el video de [Introducción a Colab](#) para obtener más información. O simplemente empieza con los pasos descritos más abajo.

Primeros pasos

El documento que estás leyendo no es una página web estática, sino un entorno interactivo denominado **cuaderno de Colab** que te permite escribir y ejecutar código.

Por ejemplo, a continuación se muestra una **celda de código** con una breve secuencia de comandos de Python que calcula un valor, lo almacena en una variable e imprime el resultado:

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
86400
```

NUMPY

NumPy



The fundamental package for scientific computing with Python

GET STARTED

D&I Grant from CZI

Including NumPy, SciPy, Matplotlib and Pandas

POWERFUL N-DIMENSIONAL ARRAYS

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

NUMERICAL COMPUTING TOOLS

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

SCIPY

 SciPy.org

SciPy.org

Install Getting started Documentation Report bugs Blogs

SciPy (pronounced "Sigh Pie") is a Python-based ecosystem of open-source software for mathematics, science, and engineering. In particular, these are some of the core packages:

 NumPy Base N-dimensional array package	 SciPy library Fundamental library for scientific computing
 Matplotlib Comprehensive 2-D plotting	 IPython Enhanced interactive console
 SymPy Symbolic mathematics	 pandas Data structures & analysis

About SciPy

- [Getting started](#)
- [Documentation](#)
- [Install](#)
- [Bug reports](#)
- [Codes of Conduct](#)
- [SciPy conferences](#)
- [Topical software](#)
- [Citing](#)
- [Cookbook](#)
- [Blogs](#)
- [NumFOCUS](#)

CORE PACKAGES:

- [NumPy](#)
- [SciPy library](#)
- [Matplotlib](#)
- [IPython](#)
- [SymPy](#)
- [pandas](#)

Search

PANDAS

 pandas

About us ▾ Getting started Documentation Community ▾ Contribute

pandas

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

[Install pandas now!](#)

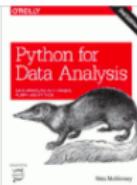
Latest version:
1.3.3

- What's new in 1.3.3
- Release date:
Sep 12, 2021
- Documentation (web)
- Documentation (pdf)
- Download source code

Follow us

[Follow @pandas_dev](#)

Get the book



With the support of:



Getting started

- Install pandas
- Getting started

Documentation

- User guide
- API reference
- Contributing to pandas
- Release notes

Community

- About pandas
- Ask a question
- Ecosystem

MATPLOTLIB



matplotlib

Version 3.4.3

Installation Documentation Examples Tutorials Contributing

home | contents » Matplotlib: Python plotting

modules | index

Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.



Matplotlib makes easy things easy and hard things possible.

Create

- Develop publication quality plots with just a few lines of code
- Use interactive figures that can zoom, pan, update...

Customize

- Take full control of line styles, font properties, axes properties...
- Export and embed to a number of file formats and interactive environments

Extend

- Explore tailored functionality provided by third party packages
- Learn more about Matplotlib through the many external learning resources

Latest stable release
3.4.3: [docs](#) | [changelog](#)

Last release for Python 2
2.2.5: [docs](#) | [changelog](#)

Development version
[docs](#)

Matplotlib cheatsheets



Support Matplotlib

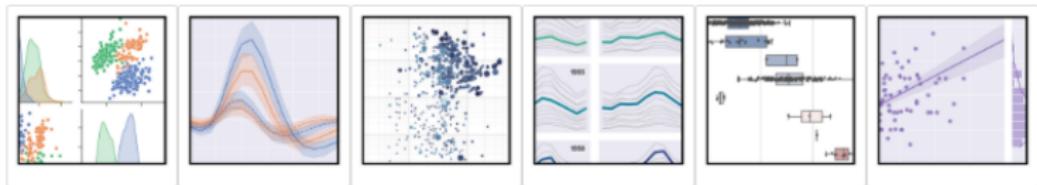
SEABORN



0.11.2

[Gallery](#)[Tutorial](#)[API](#)[Site](#) ▾[Page](#) ▾[Search](#)

seaborn: statistical data visualization



Seaborn is a Python data visualization library based on [matplotlib](#). It provides a high-level interface for drawing attractive and informative statistical graphics.

For a brief introduction to the ideas behind the library, you can read the [introductory notes](#) or the [paper](#). Visit the [installation page](#) to see how you can download the package and get started with it. You can browse the [example gallery](#) to see some of the things that you can do with seaborn, and then check out the [tutorial](#) or [API reference](#) to find out how.

To see the code or report a bug, please visit the [GitHub repository](#). General support questions are most at home on [stackoverflow](#) or [discourse](#), which have dedicated channels for seaborn.

Contents

- [Introduction](#)
- [Release notes](#)
- [Installing](#)
- [Example gallery](#)
- [Tutorial](#)
- [API reference](#)

Features

- Relational: [API](#) | [Tutorial](#)
- Distribution: [API](#) | [Tutorial](#)
- Categorical: [API](#) | [Tutorial](#)
- Regression: [API](#) | [Tutorial](#)
- Multiples: [API](#) | [Tutorial](#)
- Style: [API](#) | [Tutorial](#)
- Color: [API](#) | [Tutorial](#)

STATMODELS

The screenshot shows the statsmodels v0.13.0rc0 documentation page. At the top left is the statsmodels logo (a blue icon with three dots connected by lines) and the text "statsmodels v0.13.0rc0". To the right is a search bar with the placeholder "Search" and a magnifying glass icon. Further right is a GitHub icon and the text "statsmodels 5.6K Stars · 2.3k Forks". Below the header, the main title "statistical models, hypothesis tests, and data exploration" is displayed in white text on a dark blue background.

[statsmodels v0.13.0rc0](#)

[Installing statsmodels](#)

[Getting started](#)

[User Guide](#)

[Examples](#)

[API Reference](#)

[About statsmodels](#)

[Developer Page](#)

[Release Notes](#)



[statsmodels](#) is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration. An extensive list of result statistics are available for each estimator. The results are tested against existing statistical packages to ensure that they are correct. The package is released under the open source Modified BSD (3-clause) license. The online documentation is hosted at [statsmodels.org](#).

Scikit-Learn

[scikit-learn](https://scikit-learn.org/stable/) Install User Guide API Examples More ▾

scikit-learn
Machine Learning in Python

Getting Started Release Highlights for 0.24 GitHub

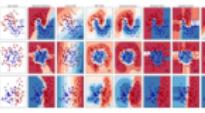
- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, and more...



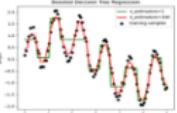
Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, and more...



Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, and more...



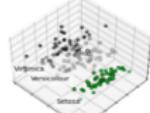
Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, increased efficiency

Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...



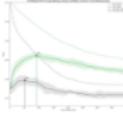
Examples

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning

Algorithms: grid search, cross-validation, metrics, and more...



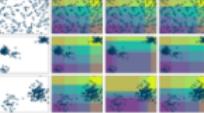
Examples

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.

Algorithms: preprocessing, feature extraction, and more...



Examples

Tensorflow - Keras

Get started Guides API docs

```
from tensorflow import keras
from tensorflow.keras import layers

# Instantiate a trained vision model
vision_model = keras.applications.ResNet50()

# This is our video-encoding branch using the trained vision_model
video_input = keras.Input(shape=[100, 100, 3])
encoded_video_sequence = vision_model(video_input)
encoded_video = layers.LSTM(128)(encoded_video_sequence)

# This is our question-encoding layer for the question input
question_input = keras.Input(shape=(100,), dtype='int32')
embedded_question = layers.Embedding(1000, 256)(question_input)
encoded_question = layers.LSTM(256)(embedded_question)

# And this is our video-question merging model
merged = keras.layers.concatenate([video, encoded_question])
outputs = keras.Dense(1000, activation='softmax')(merged)
video_qs_model = keras.Model(inputs=[video_input, question_input], outputs=outputs)
```

Deep learning for humans.

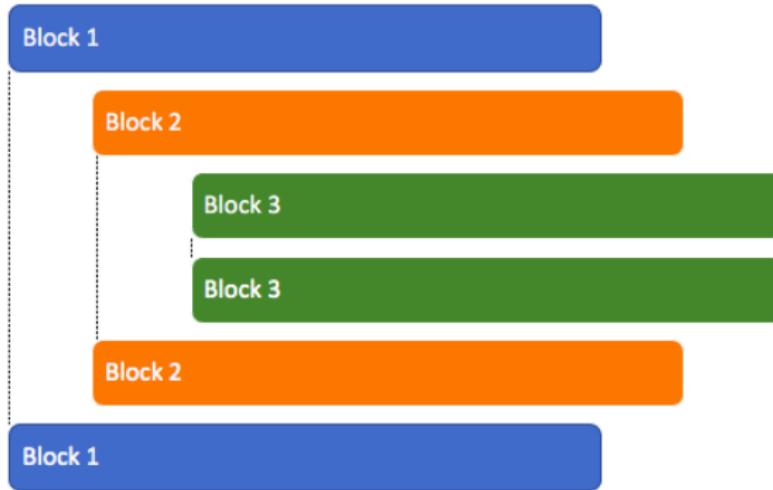
Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides.

Hello, World!!

Código de Ejemplo

```
print('Hello, World!')
```

Indentación



- Python usa como identación el espacio en blanco, en lugar de corchetes o palabras clave, para delimitar bloques.
- El tamaño de identación recomendado es de cuatro espacios.

Palabras reservadas y reglas de nombramiento

- Los nombres de las variables distinguen entre mayúsculas y minúsculas y no pueden comenzar con un número. Pueden contener letras, números y guiones bajos.

Ejemplo de Código

```
variable_Name1 = 10
```

- Palabras Reservadas: **and, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while**

Tipos de Datos

Tipo de Dato	Descripción	Ejemplos de Sintaxis
bool	Valor Booleano	True, False
int	Número entero	7
float	Número real (flotante de doble precisión)	3.1416
str	Una cadena de caracteres (string): secuencia de caracteres Unicode	'Wikipedia'

Estructura de Datos

Tipo de Dato	Descripción	Ejemplos de Sintaxis
dict	Arreglo asociativo (o diccionario) entre claves y valores; puede contener valores con tipos mezclados, las llaves deben ser inmutable (<i>hashable</i>)	{'key1': 1.0, 3: False}
list	Lista de elementos de diferentes tipos.	[4.0, 'string', True]
range	Secuencia de números que se usa comúnmente para hacer bucles un número específico de veces en el <i>for</i>	range(1, 10)
set	Conjunto desordenado, no contiene duplicados; puede contener tipos mixtos de valores inmutables	{4.0, 'string', True}
tuple	Una simple secuencia ordenada inmutable de elementos. Los elementos pueden ser de tipos mixtos, incluidos los tipos de colección.	(4.0, 'string', True)

Estructuras de Datos Especializadas

Tipo de Estructura	Toolbox	Descripción	Ejemplo de Sintaxis
array	numpy	Arreglo de elementos	<code>arr = np.array([1, 2, 3, 4])</code>
matrix	numpy	Una matriz es un arreglo bidimensional especializado que conserva su naturaleza bidimensional a través de operaciones.	<code>np.matrix([[1, 2], [3, 4]])</code>
DataFrame	pandas	DataFrame es una estructura de datos 2-dimensional etiquetadas con columnas de variables de diferentes tipos de datos.	<code>pd.DataFrame([[1, 2, 3], [4, 5, 6], [7, 8, 9]], columns=['a', 'b', 'c'])</code>

Operaciones Matemáticas

Símbolo	Significado
+	Suma
-	Resta, negativo
*	Producto
**	Potencia
/	División
//	División entera
%	Módulo

Operaciones Matemáticas

```
n1 = int(input("Número: "))
n2 = int(input("Número: "))
print("Suma = ", n1 + n2)
print("Resta = ", n1 - n2)
print("Producto = ", n1 * n2)
print("División = ", n1 / n2)
print("Potencia = ", n1 ** n2)
print("División Entera = ", n1 // n2)
print("Modulo = ", n1 % n2)
```

Operaciones Relacionales

Símbolo	Significado
<code>==</code>	igual a
<code>!=</code>	diferente a
<code><</code>	menor a
<code>></code>	mayor a
<code><=</code>	menor o igual a
<code>>=</code>	mayor o igual a

Operadores lógicos
<code>and</code>
<code>or</code>
<code>not</code>
<code>^, xor</code>

Sentencia condicional if

La sentencia condicional `if` permite que bloques de sentencias sean ejecutados dependiendo de alguna condición.

Sentencia if

```
if expression:  
    statement(s)  
elif expression:  
    statement(s)  
elif expression:  
    statement(s)  
...  
else:  
    statement(s)
```

La sentencia while

La sentencia **while** permite que un bloque de sentencias sea ejecutado repetidamente dependiendo del cumplimiento de una expresión condicional.

La sentencia while

while expression:
 statement(s)

La sentencia for

La sentencia `for` admite la ejecución repetida de una sentencia o bloque de sentencias que está controlada por una expresión iterable.

La sentencia for

```
for target in iterable:  
    statement(s)
```

Funciones

Para declarar una función solo se debe poner la palabra def seguido del nombre de la función. La función sirve para encapsular bloques de códigos.

La sentencia def

```
def function_name():
    statement(s)
```

Algoritmos

Algoritmia (David Harel 1992)

La algoritmia es más que una rama de las ciencias de la computación. Es la componente central de las ciencias de la computación, y es relevante a la ciencia, negocios y tecnología.

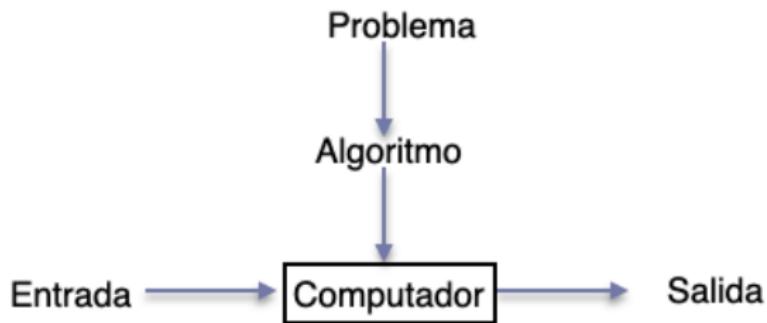
Pensamiento Algorítmico (Donald Knuth 1996)

Una persona bien capacitada en ciencias de la computación conoce cómo lidiar con los algoritmos: como construirlos, manipularlos, entenderlos y analizarlos.

Algoritmos

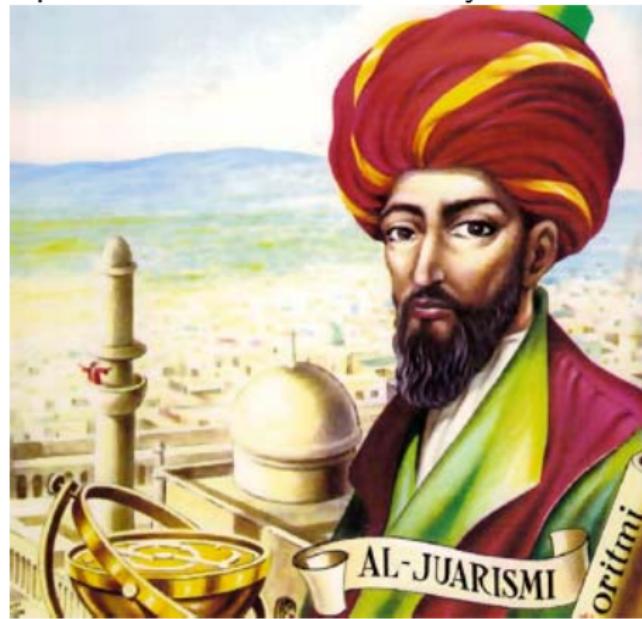
Definición de Algoritmos

Un algoritmo es una secuencia de instrucciones no-ambiguas para resolver un problema, es decir, para obtener la salida requerida para cualquier entrada válida en un tiempo finito.



Al-Juarismi

Abu Abdallah Muḥammad ibn Mūsā al-Jwārizmī (Abu Yāffar) conocido generalmente como al-Juarismi, fue un matemático, astrónomo y geógrafo persa musulmán, que vivió aproximadamente entre 780 y 850.



El 1er Algoritmo

Cómo resolver $x^2 + 10x = 39$ (Algebra)

... un cuadrado y diez raíces son iguales a 39 unidades. Entonces, la pregunta en este tipo de ecuación es aproximadamente así: cuál es el cuadrado que, combinado con diez de sus raíces, dará una suma total de 39. La manera de resolver este tipo de ecuación es tomar la mitad de las raíces mencionadas. Ahora, las raíces en el problema que tenemos ante nosotros son diez. Por lo tanto, tomamos 5 que multiplicadas por sí mismas dan 25, una cantidad que agregarás a 39 dando 64. Habiendo extraído la raíz cuadrada de esto, que es 8, sustraemos de allí la mitad de las raíces, 5, resultando 3. Por lo tanto el número tres representa una raíz de este cuadrado.

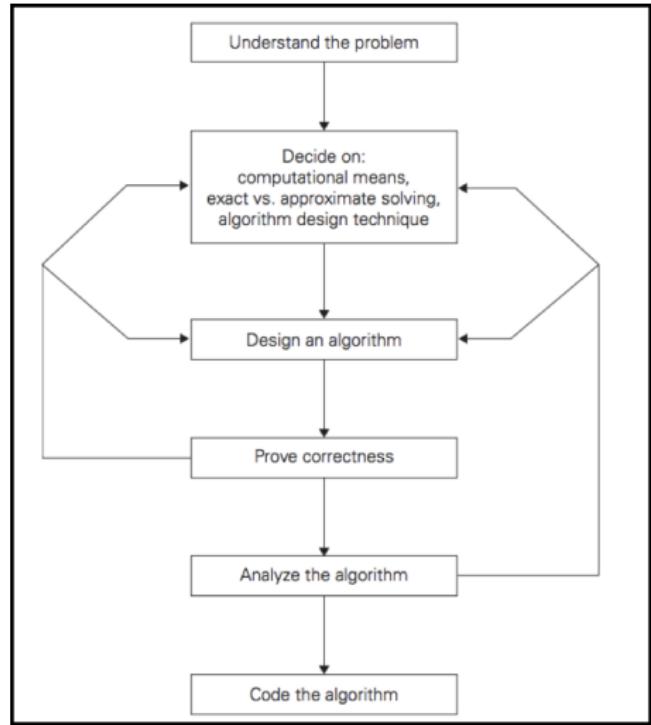


Propiedades de un Algoritmo

- Se puede representar de varias formas
- Definido (No-ambigüedad / claridad): debe ser especificado de forma rigurosa y no-ambigua.
- Entradas claramente especificadas: Entradas validas son claramente especificadas.
- Claramente especificado y genera la salida esperada: se puede demostrar que genera la salida correcta dada una entrada valida.
- Efectividad: los pasos son lo suficientemente simples y básicos.
- Finito: Debe finalizar después de un número finito de pasos. Correctitud
- Un algoritmo se puede implementar en un código de máquina de múltiples formas.

Resolución de Problemas

Un algoritmo puede ser visto como una solución procedimental de los problemas.



Ada Lovelace - 1era Programadora

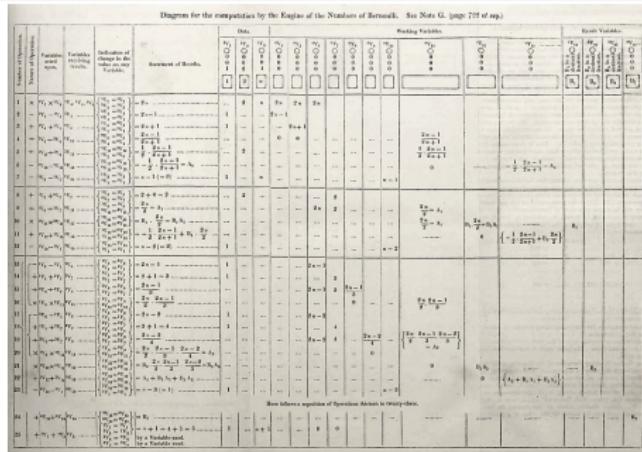
- Augusta Ada Byron, condesa de Lovelace (Londres, 1815-1852)
- Matemática y escritora británica, célebre sobre todo por su trabajo acerca de la calculadora de uso general de Charles Babbage, la denominada máquina analítica.
- Entre sus notas sobre la máquina, se encuentra lo que se reconoce hoy como el primer algoritmo destinado a ser procesado por una máquina, por lo que se la considera como la primera programadora de ordenadores.



1er Programa Informático

La nota G del algoritmo de Ada Lovelace

Calcular los números de Bernoulli mediante un salto condicional para las instrucciones repetidas en grandes cálculos que contenían muchas repeticiones en la misma secuencia de instrucciones, lo que permitiría utilizar un solo juego de tarjetas.



Números de Bernoulli

En matemáticas, los números de Bernoulli (denotados por B_n) constituyen una sucesión de números racionales con profundas conexiones en teoría de números.

$$B_m = 1 - \sum_{k=0}^{m-1} \binom{m}{k} \frac{B_k}{m-k+1} \quad (1)$$

$$B_0 = 1 \quad (2)$$

$$B^0 = 1$$

$$B^1 = 1/2$$

$$B^2 = 1/6$$

$$B^4 = -1/30$$

$$B^6 = 1/42$$

$$B^8 = -1/30$$

$$B^{10} = 5/66$$

$$B^{12} = -691/2730$$

$$B^{14} = 7/6$$

Algoritmo de Akiyama–Tanigawa

Algoritmo de Akiyama–Tanigawa

```
for m from 0 to n do
    A[m] ← 1/(m+1)
    for j from m to -1 do
        A[j-1] ← j*(A[j-1] - A[j])
return A[0] (which is Bn)
```

Taller: Programación con Python

