

Curso Intensivo Python - Día 1

Gabriel Valenzuela

@elestudianteclasses

Verano 2021

1 Agenda

- Agenda

2 Python: Introducción

- ¿Qué?¿Cuándo?¿Ah?
- Python: Introducción

3 Curso Intensivo de Python - Día 1

- Entrada y salida por consola
- Salida por consola avanzado
- Entrada por consola
- Variables
- Operaciones básicas

1 Agenda

- Agenda

2 Python: Introducción

- ¿Qué? ¿Cuándo? ¿Ah?
- Python: Introducción

3 Curso Intensivo de Python - Día 1

- Entrada y salida por consola
- Salida por consola avanzado
- Entrada por consola
- Variables
- Operaciones básicas

Agenda

- Los temas a ver el día de hoy comprenden:
 - Presentación e introducción al curso
 - Preparar el ambiente de trabajo
 - Entrada y salida por consola
 - Variables: Tipos y operaciones

1 Agenda

- Agenda

2 Python: Introducción

- ¿Qué? ¿Cuándo? ¿Ah?
- Python: Introducción

3 Curso Intensivo de Python - Día 1

- Entrada y salida por consola
- Salida por consola avanzado
- Entrada por consola
- Variables
- Operaciones básicas

1 Agenda

- Agenda

2 Python: Introducción

- ¿Qué? ¿Cuándo? ¿Ah?
- Python: Introducción

3 Curso Intensivo de Python - Día 1

- Entrada y salida por consola
- Salida por consola avanzado
- Entrada por consola
- Variables
- Operaciones básicas

Presentación e introducción al curso

Gabriel Valenzuela

- Estudiante avanzado de Ingeniería en Computación
- Desarrollador en C++
- Ayduante alumno en UNC (Análisis Matemático II)

Presentación e introducción al curso

Gabriel Valenzuela

- Estudiante avanzado de Ingeniería en Computación
- Desarrollador en C++
- Ayduante alumno en UNC (Análisis Matemático II)

¿Consultas?

- gabriel.valenzuela@mi.unc.edu.ar

Presentación e introducción al curso

Planificación

- La planificación del curso consiste en:

Fecha	Tema
06/03	Conceptos esenciales
13/03	Bucles y condicionales
20/03	Funciones, archivos y proyecto final

Presentación e introducción al curso

Planificación

- La planificación del curso consiste en:

Fecha	Tema
06/03	Conceptos esenciales
13/03	Bucles y condicionales
20/03	Funciones, archivos y proyecto final

¡Nota importante!

Fecha	Tema
06/03	Conceptos esenciales
13/03	Bucles y condicionales
20/03	Funciones, archivos y proyecto final

1 Agenda

- Agenda

2 Python: Introducción

- ¿Qué? ¿Cuándo? ¿Ah?
- Python: Introducción

3 Curso Intensivo de Python - Día 1

- Entrada y salida por consola
- Salida por consola avanzado
- Entrada por consola
- Variables
- Operaciones básicas

1 Agenda

- Agenda

2 Python: Introducción

- ¿Qué?¿Cuándo?¿Ah?
- Python: Introducción

3 Curso Intensivo de Python - Día 1

- Entrada y salida por consola
- Salida por consola avanzado
- Entrada por consola
- Variables
- Operaciones básicas

1 Agenda

- Agenda

2 Python: Introducción

- ¿Qué?¿Cuándo?¿Ah?
- Python: Introducción

3 Curso Intensivo de Python - Día 1

- Entrada y salida por consola
- Salida por consola avanzado
- Entrada por consola
- Variables
- Operaciones básicas

1 Agenda

- Agenda

2 Python: Introducción

- ¿Qué?¿Cuándo?¿Ah?
- Python: Introducción

3 Curso Intensivo de Python - Día 1

- Entrada y salida por consola
- Salida por consola avanzado
- Entrada por consola
- Variables
- Operaciones básicas

Python: Introducción - ¿Qué?¿Cuándo?¿Ah?

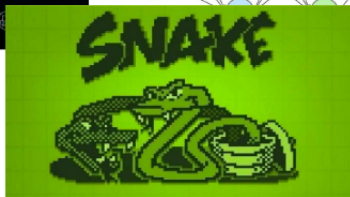
Orígenes

- Python es un lenguaje de programación **multiparadigma** e **interpretado** que surgió a finales de los años '80 por Guido van Rossum
- **Multiparadigma**: En la programación un paradigma, al igual que en el lenguaje tradicional es una “guía” para crear código, los dos más famosos son el orientado a objetos (OO) y funcional (FP) por sus siglas en inglés.
- **Interpretado**: Un lenguaje puede ser compilado (Ej: C++) o interpretado. Compilado significa que pasa por un proceso previo, la compilación, para poder ejecutar el programa. Interpretado, el intérprete va leyendo en tiempo de ejecución el programa.

→ ¿Cuál es la diferencia principal entre lenguaje compilado y ejecutado?

Python: Introducción - ¿Qué?¿Cuándo?¿Ah?

Aplicaciones



1 Agenda

- Agenda

2 Python: Introducción

- ¿Qué?¿Cuándo?¿Ah?
- Python: Introducción

3 Curso Intensivo de Python - Día 1

- Entrada y salida por consola
- Salida por consola avanzado
- Entrada por consola
- Variables
- Operaciones básicas

1 Agenda

- Agenda

2 Python: Introducción

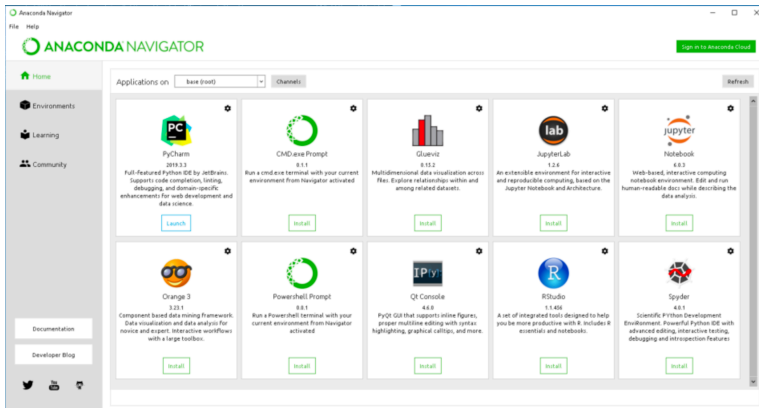
- ¿Qué?¿Cuándo?¿Ah?
- Python: Introducción

3 Curso Intensivo de Python - Día 1

- Entrada y salida por consola
- Salida por consola avanzado
- Entrada por consola
- Variables
- Operaciones básicas

Herramientas

¿Qué vamos a necesitar?



Disponible aquí

Herramientas

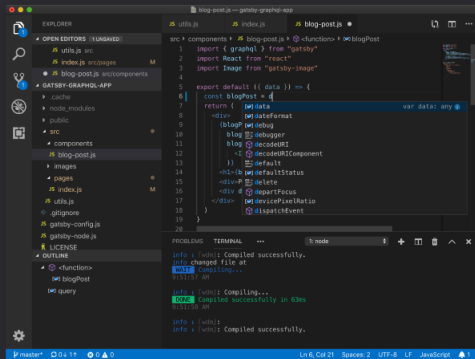
¿Qué vamos a necesitar?



Disponible aquí

Herramientas

¿Qué vamos a necesitar?



Disponible aquí

Herramientas

¿Qué podemos añadir?

The screenshot shows the Visual Studio Marketplace page for the Python extension. The header includes the Visual Studio logo and the Marketplace tab. The main section features the Python logo, the extension name 'Python' by Microsoft, and statistics: 32,302,644 installs, a 5-star rating from 368 reviews, and it is free. A description lists features like linting, debugging, IntelliSense, and Jupyter Notebook support. An installation section provides a command to install the extension and a 'Copy' button. Below the main card, there are tabs for Overview, Version History, Q & A, and Rating & Review. The 'Overview' tab is active, showing a detailed description of the extension's capabilities. On the right side, there are sections for 'Categories' (Programming Languages, Linters, Debuggers, etc.) and 'Tags' (debuggers, django, ini, etc.).

Visual Studio | Marketplace

Visual Studio Code > Programming Languages > Python

New to Visual Studio Code? [Get it now.](#)

Python
Microsoft | 32,302,644 installs | ★★★★★ (368) | Free

Linting, Debugging (multi-threaded, remote), IntelliSense, Jupyter Notebooks, code formatting, refactoring, unit tests, and more.

Installation
Launch VS Code Quick Open (Ctrl+P), paste the following command, and press enter.

`ext install ms-python.python` [Copy](#) | [More Info](#)

[Overview](#) [Version History](#) [Q & A](#) [Rating & Review](#)

Python extension for Visual Studio Code

A [Visual Studio Code extension](#) with rich support for the [Python language](#) (for all [actively supported versions](#) of the language: >=3.6), including features such as IntelliSense, linting, debugging, code navigation, code formatting, Jupyter notebook support, refactoring, variable explorer, test explorer, and more!

Quick start

- Step 1. [Install a supported version of Python on your system](#) (note: that the system install of Python on macOS is not supported).
- Step 2. Install the Python extension for Visual Studio Code.
- Step 3. Open or create a Python file and start coding!

Categories

Programming Languages Linters Debuggers
Formatters Extension Packs Data Science
Machine Learning Notebooks

Tags

debuggers django ini jinja json keybindings
linters multi-root ready pip requirements
pip-requirements python requirements.txt toml
unittest yaml

Disponible aquí

Herramientas

¿Qué podemos añadir?

The screenshot shows the Visual Studio Marketplace page for the 'Prettier - Code formatter' extension. The page has a yellow header with the extension's name and a 'Preview' badge. Below the header, there's a description of the extension as a code formatter. The 'Installation' section provides instructions on how to install it using the Visual Studio Code Quick Open command. The page also includes tabs for 'Overview', 'Version History', 'Q & A', and 'Rating & Review'. On the right side, there are sections for 'Categories' (Formatters), 'Tags' (a grid of language and framework tags), and 'Resources' (Issues).

Prettier - Code formatter Preview

Prettier | 151 installs | 0 | Free

Code formatter using prettier

Installation

Launch VS Code Quick Open (Ctrl+P), paste the following command, and press enter.

```
ext install Prettier.prettier-vscode-beta
```

[Copy](#) | [More Info](#)

[Overview](#) | [Version History](#) | [Q & A](#) | [Rating & Review](#)

Prettier Formatter for Visual Studio Code

Prettier is an opinionated code formatter. It enforces a consistent style by parsing your code and re-printing it with its own rules that take the maximum line length into account, wrapping code when necessary.

JavaScript · TypeScript · Flow · JSX · JSON
CSS · SCSS · Less
HTML · Vue · Angular
GraphQL · Markdown · YAML
Your favorite language?

[Main](#)
[passing](#)
[downloads 61M](#)
[installs 11M](#)
[code style](#)
[prettier](#)
[follow prettier 17k](#)

NOTICE: THIS PAGE AND DOCUMENTATION ARE FOR THE BETA VERSION OF THIS EXTENSION.

See [here](#) for instructions on how to install and configure the preview release.

Categories

Formatters

Tags

angular commonmark css flow formatter
graphql html ignore javascript js json jsx
less markdown md mdx multi-root ready php
prettier pug ruby scss styled-components
styled-jss swift ts typescript vue yaml yml

Resources

[Issues](#)

Disponible aquí

Herramientas

¿Qué podemos añadir?

The screenshot shows the Visual Studio Marketplace interface for the 'vscode-icons' extension. The header includes the Visual Studio logo and 'Marketplace' text. Below the header, the extension's name 'vscode-icons' is prominently displayed, followed by the publisher 'VSCode Icons Team', the number of installs (7,262,789), a star rating (4.7/5), and the fact that it is free. A circular icon representing the extension is shown to the left. The 'Installation' section provides instructions on how to install the extension using the Visual Studio Code Quick Open command. Below this, there are tabs for 'Overview', 'Version History', 'Q & A', and 'Rating & Review'. The 'Overview' tab is selected, showing a detailed view of the extension's status, including its version (1.13.0), install count, rating, and various badges for build status, dependencies, maintainability, test coverage, and issue resolution. On the right side, there are sections for 'Categories' (Themes), 'Tags' (icons, icon-theme, multi-root ready, portable mode ready, theme), and 'Resources' (Issues, Repository, Homepage, License, Changelog).

Visual Studio | Marketplace

Visual Studio Code > Themes > vscode-icons

New to Visual Studio Code? [Get it now.](#)

vscode-icons

VSCode Icons Team | 7,262,789 installs | ★★★★★ (376) | Free

Icons for Visual Studio Code

Installation

Launch VS Code Quick Open (Ctrl+P), paste the following command, and press enter.

```
ext install vscode-icons-team.vscode-icons
```

[Copy](#) | [More info](#)

[Overview](#) | [Version History](#) | [Q & A](#) | [Rating & Review](#)

vscode-icons

Visual Studio Marketplace 1.13.0 | Installs: 7.26M | Rating: average: 4.7/5 (376 ratings)

build canceled | build passing

dependencies up to date | devDependencies up to date

maintainability A | test coverage 100% | vulnerabilities 0

issue resolution 4.0 | open issues 0%

Bring icons to your [Visual Studio Code](#) (minimum supported version: 1.40.2)

supportedExtensions - vscode-icons - Visual Studio Code

Categories

Themes

Tags

icons icon-theme multi-root ready portable mode ready theme

Resources

[Issues](#) [Repository](#) [Homepage](#) [License](#) [Changelog](#)

Disponible aquí

Herramientas

¿Qué podemos añadir?

The screenshot shows the Visual Studio Marketplace page for the 'indent-rainbow' extension. The page header includes 'Visual Studio | Marketplace' and a search bar. Below the header, the extension is listed with its name 'indent-rainbow', author 'oderwat', and statistics: 1,733,418 installs, 5 stars (67 reviews), and it is free. A description states 'Makes indentation easier to read'. The 'Installation' section provides instructions to launch VS Code Quick Open (Ctrl+P), paste the command 'ext install oderwat.indent-rainbow', and press enter. A 'Copy' button and a 'More Info' link are provided. The page has tabs for 'Overview', 'Version History', 'Q & A', and 'Rating & Review'. The 'Overview' tab is active, showing a description: 'A simple extension to make indentation more readable'. There is a 'Donate' button and a note about a small donation. A code snippet shows the extension's effect on a Nim code file. On the right, there are sections for 'Categories' (Other), 'Resources' (Issues, Repository, License, Download Extension), and 'Project Details' (oderwat/vscode-indent-rainbow, 1 Pull Requests, Last Commit: 2 months ago, 8 Open Issues).

Disponible aquí

Herramientas

¿Qué podemos añadir?

The screenshot shows the Visual Studio Marketplace page for the 'Bracket Pair Colorizer' extension. The page is dark-themed and features the extension's logo (a stylized bracket with red, yellow, and blue colors) on the left. The main content area displays the extension's name, author (Coenraads), install count (4,677,223), rating (5 stars), and price (Free). Below this, there is an 'Installation' section with instructions to launch VS Code Quick Open (Ctrl+P) and paste the command 'ext install Coenraads.bracket-pair-colorizer'. A 'Copy' button and a 'More Info' link are also present. The page includes tabs for 'Overview', 'Version History', 'Q & A', and 'Rating & Review'. A section titled 'Bracket Pair Colorizer' contains an announcement about a new version being developed at <https://github.com/Coenraads/Bracket-Pair-Colorizer-2>. Below the announcement, there is a description of the extension's functionality and a 'Screenshot' section showing a code snippet with colorized brackets. On the right side, there are sections for 'Categories' (Other), 'Tags' (bracket, brackets, color, colour, multi-root ready, pair), and 'Resources' (Repository, License, Changelog, Download Extension).

Disponible aquí

Complejo vs Complicado

El Zen de Python dice:

Simple es mejor que complejo.
Complejo es mejor que complicado.

- Mientras más *complicado* es algo, más difícil de entender
- Mientras más *complejo* es algo, más partes tiene
- Los problemas complicados tienden a no tener soluciones simples
- Pero es posible **modularizar** la situación para tener muchos componentes simples
- Por ejemplo, el uso de funciones y en formas más avanzadas, clases y objetos hace que el código sea más *complejo*, pero es todavía fácil de entender

1 Agenda

- Agenda

2 Python: Introducción

- ¿Qué? ¿Cuándo? ¿Ah?
- Python: Introducción

3 Curso Intensivo de Python - Día 1

- Entrada y salida por consola
- Salida por consola avanzado
- Entrada por consola
- Variables
- Operaciones básicas

1 Agenda

- Agenda

2 Python: Introducción

- ¿Qué?¿Cuándo?¿Ah?
- Python: Introducción

3 Curso Intensivo de Python - Día 1

- **Entrada y salida por consola**
- Salida por consola avanzado
- Entrada por consola
- Variables
- Operaciones básicas

Salida por consola

- Es clásico en la programación hacer *¡Hola Mundo!*
- La operación es sencilla, pero importante ya que representa una interacción con el usuario
- Para desplegar información sin la necesidad de una GUI (Interfaz Gráfica de Usuario)

La función `print()`

- Se compone de la siguiente forma: **`print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)`**
- `*objects` puede ser una cadena, número, lista, diccionario, etc.
- El argumento **`sep`** indica el caracter que separa las palabras
- El argumento **`end`** indica el caracter que finaliza las frases.
- Los restantes argumentos vienen por defecto y no los necesitamos

¡Hola mundo!

```
1 #Imprime por consola ¡Hola mundo!  
2 #print("¡Hola mundo!")
```

1 Agenda

- Agenda

2 Python: Introducción

- ¿Qué?¿Cuándo?¿Ah?
- Python: Introducción

3 Curso Intensivo de Python - Día 1

- Entrada y salida por consola
- **Salida por consola avanzado**
- Entrada por consola
- Variables
- Operaciones básicas

Salida por consola avanzado

- Se compone de la siguiente forma: **format(value[,format_spec])**
- La función format permite convertir un valor a una expresión "formateada", que es controlado por el format_spec.
- La interpretación del format_spec dependerá el tipo de valor de argumento, sin embargo, **existe** una sintaxis estándar de formateo que es el mas comúnmente usado: Disponible aquí

Ejemplo

```
1 #Nuevo fomato de sintaxis que soporta nuevas y diferentes opciones
2 print('{0}, {1}, {2}'.format('a','b','c'))
3 #Se puede acceder por nombre de las variables
4 print('Coordendas: {lat}, {long}'.format(lat='38.23N',long='-114.810'))
5 #Salida en diferentes representaciones numéricas
6 print('int: {0:d}, hex: {0:x}, oct {0:o}, bin{0:b}'.format(16))
7 #Despliega la cantidad de cifras decimales
8 print('Valor decimal: {:.2%}'.format(1/3))
```

1 Agenda

- Agenda

2 Python: Introducción

- ¿Qué?¿Cuándo?¿Ah?
- Python: Introducción

3 Curso Intensivo de Python - Día 1

- Entrada y salida por consola
- Salida por consola avanzado
- **Entrada por consola**
- Variables
- Operaciones básicas

Entrada por consola

- La instrucción para entrada por la consola por parte de usuario consiste en la función **input()**
- **¡Ojo!** input devuelve siempre una cadena, por lo que para tomar un dato entero o flotante se "castea" con las funciones **int()** o **float()**

Ejemplo

```
1 #Toma una cadena o string de entrada
2 cadena = input('Despliego información al usuario: ')
3 #Toma un entero o int de entrada
4 entero = int(input()) #Aquí no se despliega nada
5 #Toma un flotante o float de entrada
6 flotante = float(input('Esto es opcional \n'))
```

1 Agenda

- Agenda

2 Python: Introducción

- ¿Qué?¿Cuándo?¿Ah?
- Python: Introducción

3 Curso Intensivo de Python - Día 1

- Entrada y salida por consola
- Salida por consola avanzado
- Entrada por consola
- **Variables**
- Operaciones básicas

Variables

- En Python una variable es un un nombre que se usa para hacer referencia a un valor.
- El nombre de la variable **no debe** ser una palabra reservada de Python.
- La sintaxis de las variables es:
 - nombreVariable = valorVariable
- Python es un lenguaje con tipado dinámico, esto es, las variables pueden cambiar de tipo.
- El *tipo* de la variable es una categoría del valor:
entero, flotante, cadena, lista, diccionario, objeto en general
- El tipo de una variable limita las operaciones que se pueden llevar a cabo con dichas variables.

Ejemplo

```
1 #Algunas variables
2 variableA = 10
3 variableB = 7.3
4 variableC = 'Cadena'
5 variableD = [1,2,3]
6 variableE = (1,2,3)
7 variableF = {'A':1, 'B':2, 'C':3}
8 #Imprimos los valores
9 print(variableA)
10 print(variableB)
11 print(variableC)
12 print(variableD)
13 print(variableE)
14 print(variableF)
15 #Verificamos los tipos
16 print(type(variableA))
17 print(type(variableB))
18 print(type(variableC))
19 print(type(variableD))
20 print(type(variableE))
21 print(type(variableF))
```

1 Agenda

- Agenda

2 Python: Introducción

- ¿Qué? ¿Cuándo? ¿Ah?
- Python: Introducción

3 Curso Intensivo de Python - Día 1

- Entrada y salida por consola
- Salida por consola avanzado
- Entrada por consola
- Variables
- Operaciones básicas

Operaciones básicas

■ Python

tiene numerosas operaciones que pueden llevar a cabo cálculos matemáticos.

Símbolo	Operación	Descripción
+	Adición	Suma dos números
-	Substracción	Resta dos números
*	Producto	Multiplica dos números
/	División	División convencional, el resultado es un flotante
//	División entera	El resultado de la división es un número entero
%	Modulo	Devuelve el residuo
**	Potencia	Eleva un número a un exponente

Hands On!



¿Preguntas?