introduccion.py clase01.py

```
1
2   Programming ‘Language’ = {
3
4       Introducción = [Python,
5       Micropython]
6
7
8
9
10
11  }
12
13
14
```

introduccion.py clase01.py

```
Clase 01 = {
    Presentación = [Les damos la bienvenida al curso]
}
```

Clases = {

- 👉 **Clase 01** — **Breve historia de Python y su Filosofía. Principios de diseño de Python (PEP 20). Instalación y Configuración de Python y entornos de desarrollo (IDE).**
- Clase 02 — Sintaxis Básica y Estructuras de Control. Variables, tipos de datos y operadores. Estructuras de control (if, for, while).
- Clase 03 — Estructuras de Datos. Listas, tuplas, diccionarios y conjuntos. Manipulación y métodos asociados.
- Clase 04 — Funciones y Módulos. Definición y uso de funciones. Importación y creación de módulos.

Sobre el curso {

```
    # Este curso introductorio de Python y MicroPython está
    diseñado para proporcionar a los participantes una
    introducción a la programación con Python para que los
    participantes puedan iniciarse en el emocionante mundo
    de la programación de microcontroladores con
    MicroPython. A lo largo del curso, los participantes
    explorarán y aprenderán a aplicar los conocimientos
    adquiridos en una placa del tipo ESP32. El enfoque
    principal será teórico pero con el desarrollo de
    ejercicios prácticos para finalizar con un pequeño
    proyecto electrónico.
```

}

Objetivo {

```
# Los participantes del curso podrán, finalizado el
mismo, comprender los fundamentos de la programación en
Python, entender los principios de MicroPython, conocer
sus diferencias con CircuitPython, programar la plaqueta
ESP32, explorar conectividad de redes con la placa, etc.
```

}

Régimen de cursado {

```
    # El curso requiere una dedicación total de 60 horas
    reloj, de las cuales 24 están distribuidas en 12 clases
    de 2 horas cada una y las restantes 36 es tiempo que
    cada alumno dedicara entre clases para repasar
    conceptos, practicar e investigar. Las clases del curso
    serán sincrónicas y las mismas serán los días sábados a
    partir del día 14 de septiembre en horario de 16:30 a
    18:30 (con excepción de 2 clases extras que serán día de
    semana). La modalidad será híbrida, por lo que se podrá
    asistir de forma presencial y también a distancia.
```

}

Metodología a utilizar {

```
# Las clases serán Teóricas y Prácticas. En cada sesión se impartirán conceptos para explicar los fundamentos de Python y MicroPython, utilizando presentaciones y ejemplos prácticos en vivo. Estas sesiones se complementarán con material de lectura y recursos en línea para que el estudiante pueda replicar lo aprendido en clases. El aprendizaje pretendido es orientado a proyecto, por lo que los estudiantes deberán desarrollar un proyecto en equipo de 2 o 3 personas.
```

}

Asistencia y aprobación {

Condiciones para aprobar

Asistencia al 75% de las clases (9 de 12 clases)
Aprobación: Entrega del proyecto integrador.

Se entrega **Certificado de Aprobación** a quienes hayan cumplido con el régimen de asistencia y aprobación. Aquellos que no cumplan con el régimen, recibirán un **certificado de Asistencia**.

}

‘Instructor’ Maximiliano Martin Simonazzi {

< E-Mail >
< maxisimonazzi@gmail.com >

< Github >
< www.github.com/maxisimonazzi >

< LinkedIn >
< www.linkedin.com/in/maxisimonazzi >

}

Datos importantes {

```
    # Días de cursado
    # Sábados de 16:30 a 18:30

    # Clases extras
    # 8 de Octubre de 16:30 a 18:30 - Uso de bibliotecas
    # 29 de Octubre de 16:30 a 18:30 - Git y Github

    # Modalidad
    # Presencial y virtual
```

}

Datos importantes {

Acceso al material y clases del curso

Clases presenciales
Aula 118 – Planta Baja

Link a las reuniones
bit.ly/pythonmicropython

Playlist de Youtube con las clases
www.youtube.com/playlist?list=PL9w5nSf-eLpvk-DBFcia3HqC9XlLgyCEj

Repo oficial del curso en Github
www.github.com/maxisimonazzi/introduccion-python-y-micropython-utnfrt

}

Datos importantes {

- Participantes
  - Inscriptos: 25
  - Becas otorgadas: 4

}

¿Por que Python? {

}

¿Por que Python? {

Machine Learning

< Trabajar con modelos de aprendizaje. Scikit, SciPy, etc >

Videos Juegos

< Crear videos juegos. PyGame, PyOpenGl >

App de escritorio

< Desarrollar aplicaciones de escritorio. tkinter, PyQT, etc >

Seg. Informática

< Programar scripts de pruebas y análisis automático de vulnerabilidades y P.T. >

}

¿Por que Python? {

Desarrollo Web

< Aplicaciones web FullStack usando como backend Flask, Django, SQLAlchemy, etc >

Testing y QA

< Automatización de testing y funcionalidades. Selenium, beautifull soup, etc >

Big Data

< ETL, Extraer, Procesar, Almacenar y Analizar datos. Pandas, NumPy, MatPlotLib, etc >

Electrónica

< Programación de microcontroladores ESP32, Raspberry Pi Pico, STM, etc >

}

¿Qué es Python? {

```
# Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma. Es un lenguaje interpretado, dinámico y multiplataforma

# Administrado por Python Software Foundation, posee una licencia de código abierto, denominada Python Software Foundation License. Python se clasifica constantemente como uno de los lenguajes de programación más populares
```

}

¿Qué es Python? {

Multiparadigma

< Soporta programación imperativa, estructurada, POO, funcional, etc >

Multiplataforma

< Su código puede correr en Windows, Linux, Mac, Microcontroladores, etc. Se puede reusar codigo >

Tipado Fuerte y Dinamico

< El tipo de las variables se decide en tiempo de ejecución. No se permite operar tipos diferentes >

Interpretado

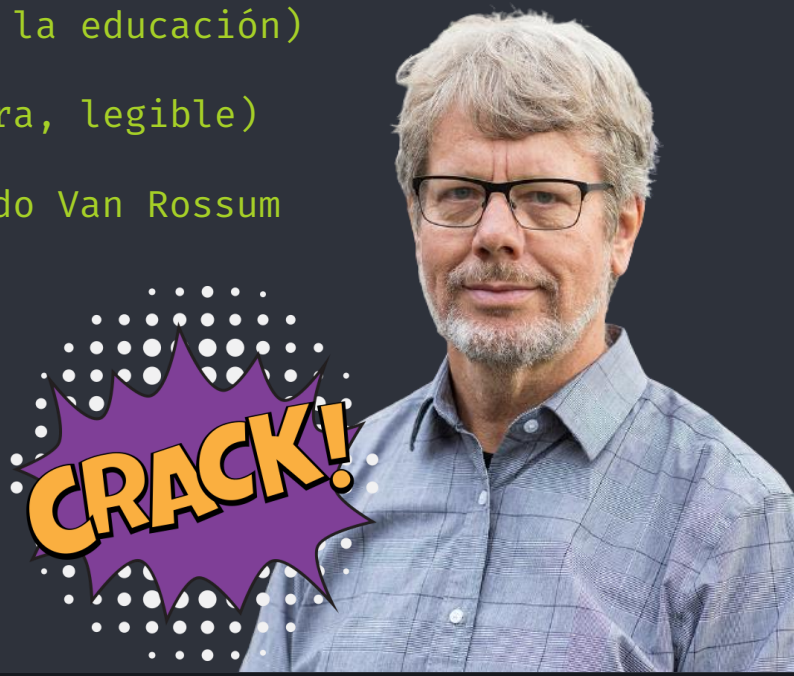
< El código no se compila a lenguaje máquina, sino que se ejecutan las instrucciones a medida que se las lee >

}

Sobre Python {

```
    # Fácil de aprender (alto impacto en la educación)
    # Alto nivel (sintaxis sencilla, clara, legible)
    # Creado a finales de los 80’por Guido Van Rossum
    # Completamente libre, Open Source
    # Amplia librería standard
    # Pagina Oficial: www.python.org
```

}



Sobre Python {

```
    # Sucesor espiritual del lenguaje ABC

    # El nombre proviene del programa humorístico
    británico Monty Python

    # El desarrollo comienza en 1989

    # 20 de Febrero de 1991 publica el código con v.0.9.0

    # En Enero 1994 se publica la v.1
```

}

Sobre Python {

```
# En el 2000 se lanza Python v.2

# En el 2008 se lanza Python v.3 ¡¡¡Liberen a la bestia!!!

# La transición de v.2 a v.3 fue lenta por sus diferencias

# A día de hoy, nos encontramos en la v.3.12.6

# La demanda laboral no para de crecer

# PEP-8 (estilo) y PEP-20 (zen) son las guías principales.
```

}

Interpretado vs Compilado {

```
# Los lenguajes compilados utilizan un compilador que procesa el programa completo en una sola vez, generando un código intermedio o código máquina. Estos lenguajes, como C, C++, y C#, ofrecen una eficiencia notable, ya que los errores se identifican después de la verificación completa del programa, asegurando una mayor integridad del código
```

}

Interpretado vs Compilado {

```
# Los lenguajes interpretados adoptan un enfoque más lineal mediante el uso de un intérprete que ejecuta el programa línea por línea. Lenguajes como JavaScript y Python pertenecen a este grupo. Aunque son menos eficientes que los lenguajes compilados, la facilidad de depuración y la interpretación línea por línea hacen que sean una opción atractiva
```

}

Interpretes {

CPython

Micropython

Circuitpython

}

Interpretes {

CPython

< CPython es la implementación mas usada del interprete y su lenguaje. Decimos entonces que corremos Python **en** CPython. >

< Esta escrito en C (por eso la C del nombre) y es la implementación de referencia >

< Tiene Garbage Collector, manejo dinámico de memoria, manejo de errores, debugger, etc >

< El GIL (Global Interpreter Lock) permite ejecutar un hilo principal con el bytecode de Python en un proceso único. Es por ello que con Python tenemos soporte para programación asíncrona pero no programación en paralelo >

}

Interpretes {

Micropython

< Micropython es una implementación de un interprete pero diseñada y optimizada específicamente para funcionar en entornos con recursos limitados (como en microcontroladores). Su huella es menor a 2 Mb. >

< Se escribe en muchos foros de comunidad como μPython >

< A diferencia de CPython, no requiere un S.O. para trabajar sino que funciona como si fuese el S.O. >

< Al igual que CPython esta escrito en C. >

}

Interpretes {

Circuitpython

< Circuitpython es un fork de Micropython mantenido y llevado adelante por la empresa Adafruit para darle soporte a todas sus placas y diseños. >

< Las diferencias entre Circuitpython y Micropython son prácticamente nulas. Se reduce solo a dar mejor soporte a placas de la empresa, es por ello que los códigos y librerías de ambos pueden ser intercambiados y reutilizados. >

< Ambos tienen una gran comunidad lo cual es bueno. Micropython opera casi exclusivamente en github mientras que Circuitpython en discord y su foro >

}

```
1
2
3  Aprender a programar
4  es aprender a pensar.
5
6
7
8
9
10
11 { Steve Jobs; }
12
13
14
```


1
2
3
4
5
6
7
8
9
10
11
12
13
14

{ Nos vemos en la proxima clase }