

# ALGORITMOS Y PROGRAMACIÓN

**Licenciatura en Matemática Aplicada**

SOBRE ESTA  
ASIGNATURA

# OBJETIVOS DE LA MATERIA

- Asimilar **conceptos fundamentales** de programación imperativa mediante la ejercitación. 🚴
- Familiarizarse con **técnicas** de programación. 🔧
- Incorporar **buenas prácticas** de programación. 📁
- Reconocer la necesidad de **comprobar el buen funcionamiento** de los programas. 🔍
- Adquirir **habilidad** en el abordaje computacional de problemas numéricos simples. ❌ ÷

# PROGRAMACIÓN ¿IMPERATIVA?

Existen distintos enfoques, “paradigmas”, o **Modelos de Programación** (asignatura de 5to año)

- Imperativa (se estudia en esta asignatura)
- Orientada a objetos
- Funcional
- Lógica
- Otras
  - Dirigida por eventos
  - Orientada a aspectos

# CONTENIDOS MÍNIMOS

Introducción a la programación imperativa. Entrada/Salida. Estado, variable y asignación. Condicional. Iteración. Variante e invariante. Programación estructurada. Subalgoritmos: bloques, funciones y procedimientos. Pasaje de parámetros. Recursión. Diseño top-down y bottom-up. Abstracción. Tipos elementales y estructurados. Especificación e implementación. Correctitud. Verificación. Programación de algoritmos elementales de teoría de números.

# METODOLOGÍA DE APRENDIZAJE

- Clases eminentemente prácticas.
- Secuencia de ejercicios.
- Dificultad incremental.
- Introduciendo **conceptos** gradualmente 🌑🌒🌓🌔🌕.
- Interrumpiendo la ejercitación para discutirlos, señalar buenos y malos **hábitos**.
- La ejercitación ayudará a fijar los conceptos, reconocer los vicios y adquirir buenas técnicas.

# DINÁMICA DE LAS CLASES

Trataremos de hacer bloques cortos

- Pueden realizar preguntas en cualquier momento
  - interrumpiéndome
  - levantando la mano
- Los bloques cortos tendrán brevísimas pausas para levantarse, caminar un poco o quedarse charlando

# REGULARIDAD

Artículo 5° del Anexo I de la Resolución CD N° 73/2021.

- estar inscripto en la materia como alumno/a regular
- aprobar al menos el 60 % de los trabajos prácticos (6 de 10 tareas)



# PROMOCIÓN

Artículo 7° del Anexo I de la Resolución CD N° 73/2021.

- estar inscripto en la materia como alumno/a regular
- haber aprobado el cursillo de nivelación
- aprobar las tareas o trabajos prácticos (todos), con nota no inferior a 6
- aprobar un coloquio

Aparte:

- El promedio de los trabajos prácticos debe ser  $\geq 7$ .

# APROBACIÓN (EN CASO DE NO PROMOCIONAR)

Se tomará un examen que contendrá

1. una parte de preguntas conceptuales,
2. otra de resolución con la computadora
3. y un coloquio.

La parte (2) podrá consistir de una tarea a realizar previamente y otra durante el examen.

# SOBRE LOS COLOQUIOS

Los coloquios (tanto en la promoción como en el examen) se enfocan en las soluciones presentadas, posibles variantes y preguntas conceptuales.

# LOS DOCENTES

- Alejandro Tiraboschi, 9:00 - 10:45
- Javier Lezama, 11:00 - 12:45

# COMUNICACIÓN SOBRE LA ASIGNATURA

**Aula virtual:** próximamente (no será la herramienta principal)

**Sala “Estudiantes de AyP” de Google chat:**

<https://mail.google.com/chat/u/0/#chat/space/AAAAbmdAroY>

**Classroom “Algoritmos y Programación / 2022-1”:** por invitación.

**Mail de los profes:** [alejandro.tiraboschi@unc.edu.ar](mailto:alejandro.tiraboschi@unc.edu.ar),  
[javier.lezama@unc.edu.ar](mailto:javier.lezama@unc.edu.ar)

i PAUSAAA !

APRENDER A  
PROGRAMAR

≠

APRENDER UN  
LENGUAJE

# APRENDER PROGRAMACIÓN ≠ APRENDER UN LENGUAJE

- Aprender programación no es lo mismo que aprender un lenguaje de programación.
- Pero para aprender a programar necesitamos ejercitar.
- Para ello elegiremos un lenguaje de programación: Python.
- Necesitaremos ir aprendiendo **particularidades** de Python.
- Importante mantener el foco en lo **conceptual** de la programación imperativa.
- Habrá un conflicto constante entre lo **conceptual** y lo **particular**, estemos atentos para separar uno del otro.



# ¿POR QUÉ PYTHON?

Existen numerosos lenguajes de programación imperativa: C, C++, Java, JavaScript, ... ¿por qué Python?

- [Python: 7 Important Reasons Why You Should Use Python](#)
- [5 reasons you should learn Python now | Pluralsight](#)
- [Why Learn Python - Best Programming Language](#)
- [Why Python is considered the top programming language ahead of JavaScript and C++](#)
- [6 Reasons Why Python Is the Programming Language of the future](#)

Con Python, mientras aprendemos a programar adquirimos una herramienta que es aplicable a numerosos campos.

# ¿POR QUÉ PYTHON?

Existen herramientas muy convenientes para aprender Python:

**Jupyter Notebook:** nos permite escribir texto enriquecido y programar y ejecutar programas Python en un “cuaderno” interactivo.

**Google Colab:** una implementación de los cuadernos Jupyter en la nube. No hay necesidad de instalar nada en nuestras computadoras. Si estás logueado en una cuenta de Google podés empezar a trabajar <https://colab.research.google.com/>

# BIBLIOGRAFÍA SOBRE PYTHON

Hay abundante material disponible, por ejemplo:

- Introduction to programming using Python. Daniel Liang. Armstrong Atlantic State University, 2013.
- [Tutorial de Python](#) – documentación de Python – 3.9.1
- [Python Practice Book](#), 2019, Anand Chitipothu (creative commons).

# BIBLIOGRAFÍA SOBRE PYTHON (2)

- Introduction to Computation and Programming Using Python, MIT Press, 2013, John V. Guttag
- Think Python, 2012 o 2016, Allen B. Downey.
- Innumerables recursos: [python.org](http://python.org), [python.org.ar](http://python.org.ar) (material en castellano)

Como todo lenguaje, se aprende a través de su uso.

Consultar la bibliografía del curso en el aula virtual.

# APRENDIENDO UN LENGUAJE DE PROGRAMACIÓN

- Aprendiendo los fundamentos.
- Escribiendo programas básicos y aumentándoles progresivamente la dificultad.
- Ejecutando, mirando y modificando programas escritos por nosotros y por otros.
- Ante la duda sobre qué hace un cierto programa, o un fragmento de un programa, intentar descubrirlo modificándolo, ejecutándolo.
- Consultar los manuales del lenguaje, consultar internet.
- Consultar con un compañero o con profes o el foro.

# OBJETIVOS DE LA MATERIA

- Enseñarles **conceptos fundamentales** de programación imperativa.
- Proporcionarles **habilidad** en el uso de Python para resolver problemas numéricos, en lo posible relacionados con los contenidos previos o impartidos en otras asignaturas de este cuatrimestre.
- Prepararles para las **materias siguientes**, en particular para Análisis Numérico I y II (2º año) y Algoritmos y Estructura de Datos (3er año).

EMPECEMOS A  
PROGRAMAR

# EMPEZAMOS JUGANDO

Programar tiene algo de juego, empecemos a programar jugando:

En [code.org](https://code.org) encontramos muchos ejercicios de programación, haremos parte de uno de sus cursos: [el curso acelerado](#)!

<https://studio.code.org/s/20-hour>



Ejemplo: El laberinto: [ejercicio 1](#).

Resolver hasta el ejercicio 5. Si terminás muy rápido, pensá qué otra cosa se te ocurre hacer en esos ejercicios.



# ¿QUÉ ES PROGRAMAR?

En lenguajes de programación imperativos, programar

- es dar una **secuencia de instrucciones**
  - “Preparación:
    - Pela las bananas y en un recipiente tritúralas con un tenedor.
    - Mezcla la crema de leche con el chocolate untable.
    - Añade el "puré de banana" a la mezcla de chocolate.
    - Reserva en el refrigerador por una hora y disfruta.”
- a un robot/**una máquina** que sabe ejecutarlas


**Es importante conocer qué instrucciones sabe ejecutar ese robot/esa máquina. ¿nuestro caso?**

# INSTRUCCIONES QUE SABE EJECUTAR




- avanzar
- girar a la derecha ↻ ▼ girar a la izquierda ↺ ▼

# PRIMERAS PREGUNTAS

- ¿Qué pasa cuando  choca contra una pared?
- ¿Da lo mismo cualquier pared?
- ¿Qué pasa si no se elige la solución óptima?
- ¿Qué significa solución óptima?
- ¿Qué otras soluciones hay?
- ¿Cuántas soluciones hay?
- ¿Qué ocurre si se pasa de largo?
  - ¿Por qué ocurre eso?
  - ¿Qué les parece esa decisión?

# ¿ALGUNA QUEJA? (POR EJEMPLO EN EL EJERCICIO 5)



Tuvimos que pegar 3 veces seguidas el bloque   
¿Y si la distancia entre los personajes fuera mayor?

# SIGAMOS JUGANDO

Estaría buenísimo no estar obligado a escribir varias veces seguidas la misma instrucción: a partir de ahora la máquina sabe **repetir** instrucciones!

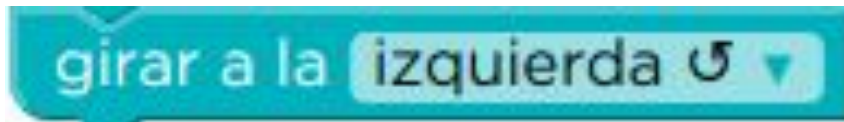
Resolver hasta el ejercicio 9.

Si terminás muy rápido, pensá qué otra cosa se te ocurre hacer en esos ejercicios. Aprovechá para mirar el programa que genera seleccionando “Mostrar el código”.

# INSTRUCCIONES QUE SABE EJECUTAR



- Instrucciones simples:



- Instrucciones complejas:



# CICLOS

- La posibilidad de **repetir** (también se dice “**iterar**”) instrucciones es común a todos los lenguajes de programación imperativos.
- Se llaman **ciclos**, **bucles**, o **loops** en inglés.
- ¿Qué es una solución óptima ahora?

# CICLOS CONDICIONALES

- Hay de distintos tipos de ciclos, uno de ellos es el que repite un **número determinado** de veces.
- Otro muy importante es el que repite hasta que se cumpla una **cierta condición**, hasta que algo ocurra.
- Éstos se llaman **ciclos condicionales**.
- Resolver hasta el ejercicio 13 de laberintos.
- No olvides mirar el programa que se genera seleccionando “Mostrar el código”.



# CICLOS PELIGROSOS

Los ciclos-hasta-que-se-cumpla-una-cierta-condición,  
introducen un nuevo riesgo. 🦖 ⚠️ 💣 💀

¿Cuál?

¿Podés hacer un programa en el que eso ocurra?

# INSTRUCCIONES QUE SABEN EJECUTAR



Y



- Instrucciones simples:

girar a la derecha ↻ ▼

girar a la izquierda ↻ ▼

avanzar

- Complejas:

repetir 5 veces

haz

repetir hasta



haz

# INSTRUCCIONES CONDICIONALES

- Hemos visto las repeticiones hasta que se cumple una **cierta condición**,
- Con frecuencia es conveniente actuar de cierta manera solamente si se cumple una cierta condición (ejemplo: si hay o no un camino a la izquierda).



- Las llamamos **instrucciones condicionales**.
- Resolver hasta el ejercicio 17.

# INSTRUCCIONES QUE SABEN EJECUTAR



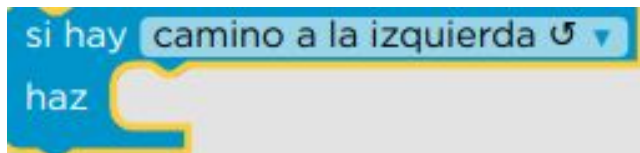
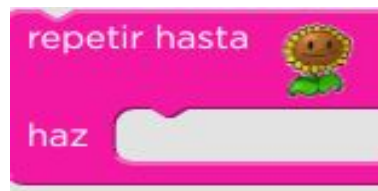
Y



- Instrucciones simples:



- Complejas:



# INSTRUCCIONES CONDICIONALES (2)

- Las instrucciones condicionales se pueden utilizar más generalmente para realizar dos acciones diferentes según la condición se cumpla o no.



- Resolver hasta el ejercicio 20.

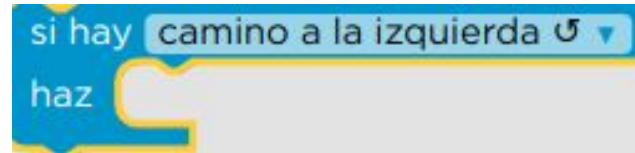
# INSTRUCCIONES QUE SABEN EJECUTAR Y



- Instrucciones simples:



- Complejas:



**“derecha” “5” son valores de parámetros**

# REPASANDO LOS CONTENIDOS MÍNIMOS

**Introducción a la programación imperativa.** Entrada/Salida. Estado, variable y asignación. **Condicional. Iteración.** Variante e invariante. **Programación estructurada.** Subalgoritmos: bloques, funciones y procedimientos. Pasaje de parámetros. Recursión. Diseño top-down y bottom-up. Abstracción. Tipos elementales y estructurados. Especificación e implementación. Correctitud. Verificación. Programación de algoritmos elementales de teoría de números.