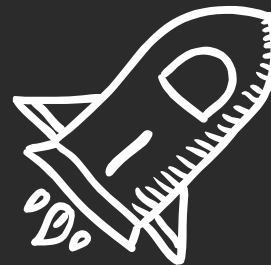




Semana 1

¡Bienvenidos!



¿Qué trataremos esta semana?



Durante esta primer semana, abordaremos conceptos básicos necesarios para la comprensión de este curso. Además, te ayudaremos a instalar las herramientas necesarias para cursar correctamente esta etapa.

Temas de hoy

1 Presentación del curso
y objetivos

3 Ejemplos de problemas
lógicos y su abordaje

2 Conceptos básicos

4 Instalación de
herramientas

1

Presentación y objetivos

Te damos la bienvenida a este curso de desarrollo web y base datos, donde realizaremos un recorrido de aprendizaje desde las bases y fundamentos de la programación usando la lógica para que así, puedas adquirir un pensamiento crítico y encontrar soluciones a problemas, e implementes todo el aprendizaje en un proyecto final integrador, donde desarrollarás una página web.

2

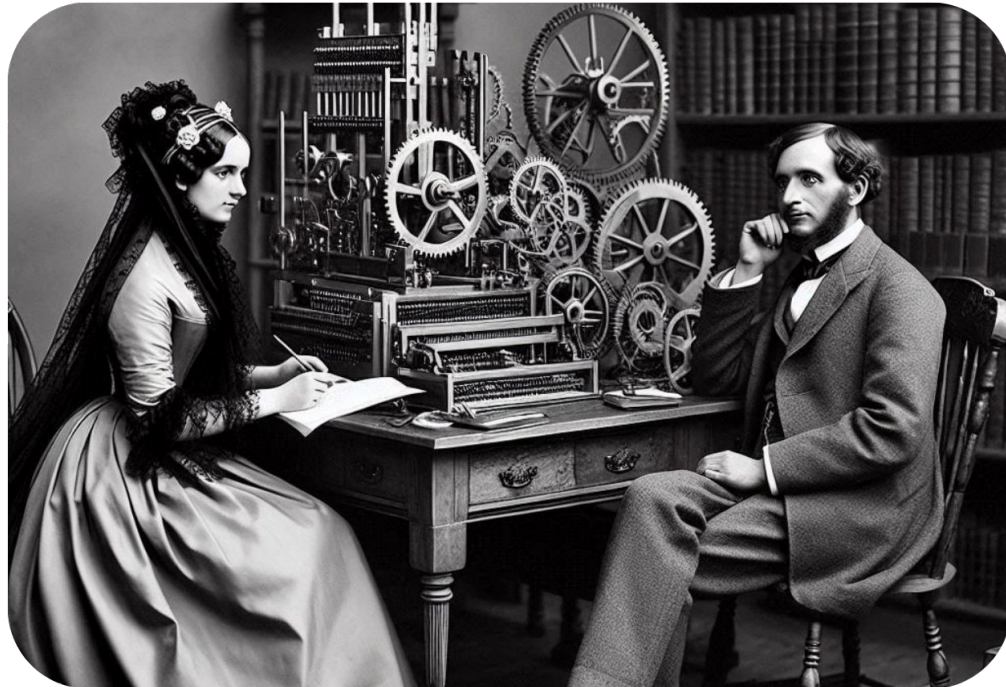
Conceptos básicos

Para que comprendas mejor desde el comienzo, te vamos a dar algunos conceptos básicos de manera introductoria.

Lógica de programación

Una de las cosas básicas que todo buen programador debe adquirir, es el pensamiento lógico para resolver y encontrar soluciones a problemas planteados.

Cuando uno adquiere esta base/lógica puede ser capaz de resolver problemas de manera eficiente, escalonada y flexible. Adquirir este pensamiento es fundamental para tener un pensamiento crítico al momento de encontrar soluciones.



Representación generada con IA de Ada Lovelace y Charles Babbage

Algoritmo

Usamos algoritmos cotidianamente pero sin saberlo. Cada vez que aprendemos algo nuevo, seguimos una serie de pasos para poder lograr obtener un resultado. Como lo es una receta de cocina por ejemplo, en la cual hay una serie de pasos a realizar de manera secuencial. Siguiendo esta serie de pasos o instrucciones, logramos obtener el resultado esperado.

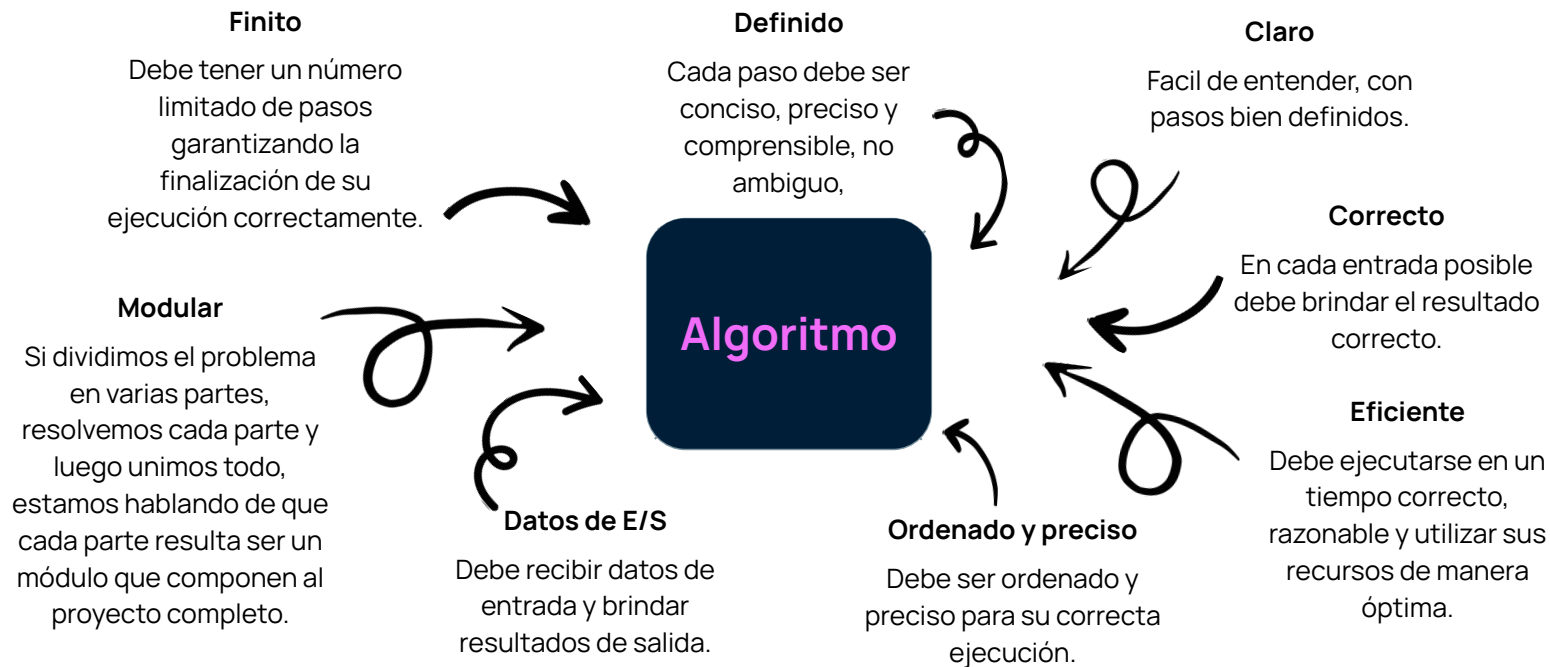
En la programación, una vez planteado el problema, usamos algoritmos como una secuencia de pasos bien definidos para resolver ese problema.

Estos algoritmos pueden ser expresados de forma escrita, representados en pseudocódigo o diagramas de flujo. Una vez confeccionado esto, se puede pasar a cualquier lenguaje de programación para realizar la ejecución de la solución.



Representación generada con IA de Ada Lovelace y la Máquina Analítica

Características de un buen algoritmo



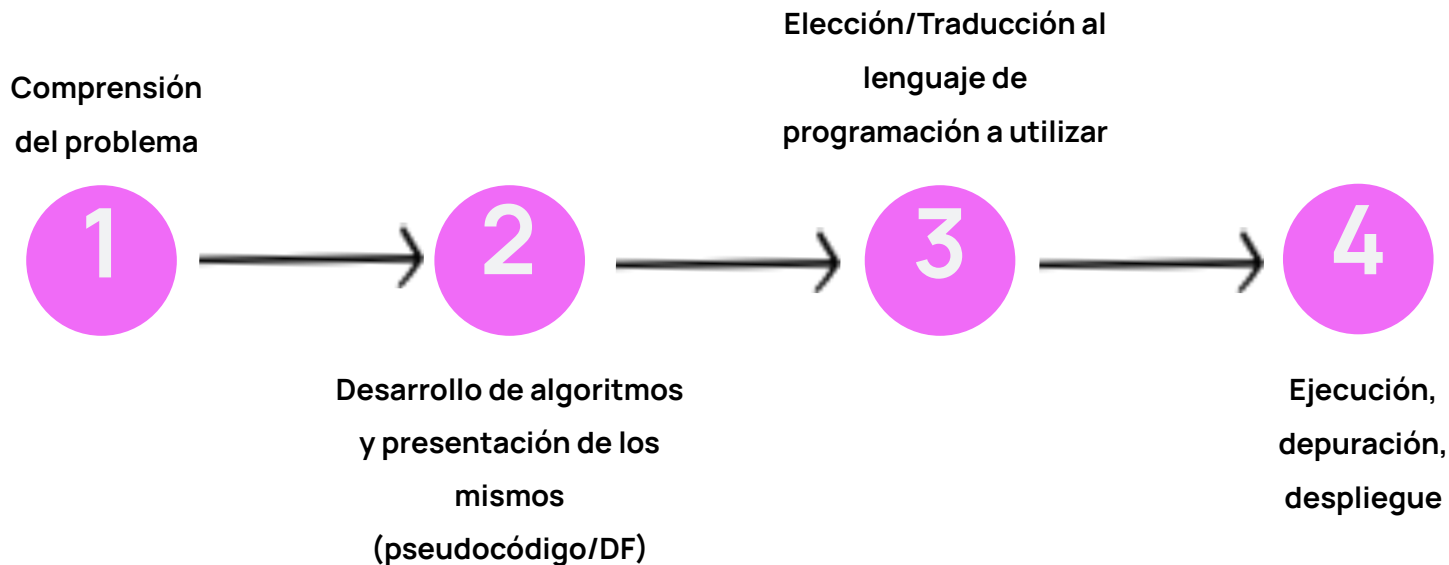
¿Qué pasos debo seguir para resolver un problema?

Comprensión del problema	Desarrollar algoritmos	Presentación del algoritmo	Traducir al lenguaje de programación a usar
Muchas veces intentamos resolver un problema cuando se nos presenta, sin comprenderlo completamente, por esto, es crucial comprender bien cuál es el problema, dividirlo en partes pequeñas y manejables para encontrar una solución óptima,	Una vez comprendido completamente el problema y dividido, se pueden identificar etapas para su resolución. Para cada etapa se puede crear un algoritmo, donde cada etapa irá resolviendo un problema específico que llevará a la solución completa.	Una vez que se tenga el algoritmo o los algoritmos, los podemos pasar a pseudocódigo o diagrama de flujo. Esto nos ayudará a tener la secuencia correcta de la programación para el siguiente paso.	Otra parte importante, luego de que se llegó a la solución del problema, es elegir el lenguaje de programación, ya que cada lenguaje tiene su particularidad y cada proyecto puede tener requisitos específicos. Con respecto a este punto, no te preocupes que la experiencia y el tiempo te darán la comprensión necesaria para saber que lenguaje usar en cada proyecto.

Problema resuelto y... listo para ejecutar

Ejecución y depuración	Proyecto listo para ser lanzado	Mantenimiento sostenible y escalable del proyecto
<p>Cuando comenzamos con las primeras ejecuciones, se dará, que se deberán resolver y ajustar algunas cosas. Esta etapa, generalmente pasará por el área de testeo, para volver a la etapa de desarrollo y solucionar posibles fallos.</p>	<p>Luego de ajustar y afinar detalles que surjan durante la parte del testeo o control de calidad, el proyecto estará listo para ser desplegado y lanzado al usuario final.</p>	<p>Algo que se debe tener en cuenta, es que el proyecto debe ser capaz de poder mantenerse y modificarse en el tiempo. Muchas veces el producto final está bien en principio, pero en determinado momento puede crecer, o migrar a otro sistema, por lo que éste deber ser escalable y cumplir con lo requerido en un posible futuro.</p>

Repasando las etapas del proyecto



3

Ejemplo de problemas y cómo abordarlos

En las siguientes diapositivas veremos un problema simple. Lo comprenderemos y haremos una serie de pasos para abordarlo, resolverlo y llegar a un resultado final.

Ejemplo de un problema simple:

Problema: Verificar si una persona es mayor de edad una vez que ingresa su año de nacimiento.

- ▶ Siguiendo los pasos aprendidos anteriormente, abordaremos el problema propuesto, desde la comprensión del problema y la creación del algoritmo, hasta dejarlo listo para su uso.
- ▶ **Algoritmo:**
 - *Pedir y leer el año de nacimiento de la persona.
 - *Calcular la edad:
Restar el año actual con el año de nacimiento ingresado.
 - *Verificar si la persona es mayor o menor.
 - *Mostrar el resultado.
- ▶ **Presentación en Pseudocódigo:**


```
Algoritmo calcular_edad
Definir anio_actual, anio_nac, calculo Como Entero
anio_actual = 2024
Imprimir 'Ingrese su año de nacimiento: '
Leer anio_nac
calculo = anio_actual - anio_nac
Si calculo >= 18 Entonces
    Escribir 'Usted es mayor de edad.'
SiNo
    Escribir 'Usted es menor de edad.'
FinSi
FinAlgoritmo
```
- ▶ **Traducción a Python (como lenguaje de programación):**

```
anio_actual = 2024
anio_nac = int(input('Ingrese su año de nacimiento: '))
calculo = anio_actual - anio_nac
if calculo >= 18:
    print('Usted es mayor de edad.')
else:
    print('Usted es menor de edad.')
```

Ejemplo de un problema simple:

Problema: Verificar si una persona es mayor de edad una vez que ingresa su año de nacimiento.

- ▶ Una vez resuelto el problema con su algoritmo, presentado en pseudocódigo y traducido a un lenguaje de programación, en este caso Python, ya lo podemos ejecutar y probar.

- ▶  1.py > ...

```
1  anio_actual = 2024
2  anio_nac = int(input('Ingrese su año de nacimiento: '))
3  calculo = anio_actual - anio_nac
4  if calculo >= 18:
5      print('Usted es mayor de edad.')
6  else:
7      print('Usted es menor de edad.')
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```
PS C:\Users\Pc\Documents\Info> & C:/Users/Pc/AppData/Local/Prog
Ingrese su año de nacimiento: 2020
Usted es menor de edad.
PS C:\Users\Pc\Documents\Info> 
```

- ▶ Habiendo ejecutado el programa varias veces y probándolo bien, este programa se encuentra listo para ser lanzado como producto final.

Este programa base, a medida que se requieran otras acciones, se lo puede ir mejorando y escalando, por lo que es un programa sostenible y escalable a futuro.

Mejorando nuestro ejemplo anterior

Problema: Verificar si una persona es mayor de edad una vez que ingresa su año de nacimiento.

- ▶ Si quisiéramos mejorar nuestro programa anterior de ejemplo, podríamos usar otros recursos disponibles, como el uso de funciones, módulos y sentencias para manejo de errores. Por ejemplo:

```
12.py > ...
1  import datetime
2
3  def verificar_mayoria_edad():
4      try:
5          anio_nac = int(input("Ingresa tu año de nacimiento: "))
6          anio_actual = datetime.datetime.now().year
7          edad = anio_actual - anio_nac
8
9          if edad >= 18:
10             print("Usted es mayor de edad.")
11         else:
12             print("Usted es menor de edad.")
13     except ValueError:
14         print("Ingresa un año válido.")
15
16     verificar_mayoria_edad()
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```
PS C:\Users\Pc\Documents\Info> & C:/Users/Pc/AppData/Local/Programs/Python/Python39-32/Python.exe 12.py
Ingresa tu año de nacimiento: 2020
Usted es menor de edad.
PS C:\Users\Pc\Documents\Info> & C:/Users/Pc/AppData/Local/Programs/Python/Python39-32/Python.exe 12.py
Ingresa tu año de nacimiento: abc
Ingresa un año válido.
PS C:\Users\Pc\Documents\Info> & C:/Users/Pc/AppData/Local/Programs/Python/Python39-32/Python.exe 12.py
Ingresa tu año de nacimiento: 2000
Usted es mayor de edad.
PS C:\Users\Pc\Documents\Info> 
```

- ▶ Como se puede observar, ya hemos escalado el programa para obtener el año actual de forma automática y además, se están manejando los datos de ingreso, como en el ejemplo que no se ingresa correctamente un año, por lo que el programa puede manejar esto sin dar errores y brindar una leyenda donde solicita que se ingrese un año correctamente.



Conclusión final

Durante este ejemplo, pudimos abordar un problema muy simple, pero que a medida que se requiera, se puede ir escalando. Incluso, este puede ser un módulo para un programa más complejo, como por ejemplo, puede formar parte de un sistema de votos, donde se puede verificar la edad apta para votar. Esta es la importancia de la división de un problema en problemas pequeños. Teniendo varios módulos, que conforman el proyecto completo, podemos detectar fallas mucho más rápido y de manera eficiente, e ir al módulo en el que está ocurriendo la falla para poder resolver de manera rápida y sin tener que recorrer todo el código completo.

4

Instalación de herramientas

En este curso, utilizaremos diversas herramientas para llevar a cabo todas las actividades programadas. Para facilitar la instalación y el uso de estas herramientas, hemos creado videos tutoriales detallados. Puedes encontrar estos videos en el campus virtual

PSeint

Además del video de instalación, te vamos a dejar los pasos para la instalación de PSeint.

1 - Ingresá a

<https://pseint.sourceforge.net/index.php?page=descargas.php>

2 - Descargá la versión compatible con tu sistema operativo.

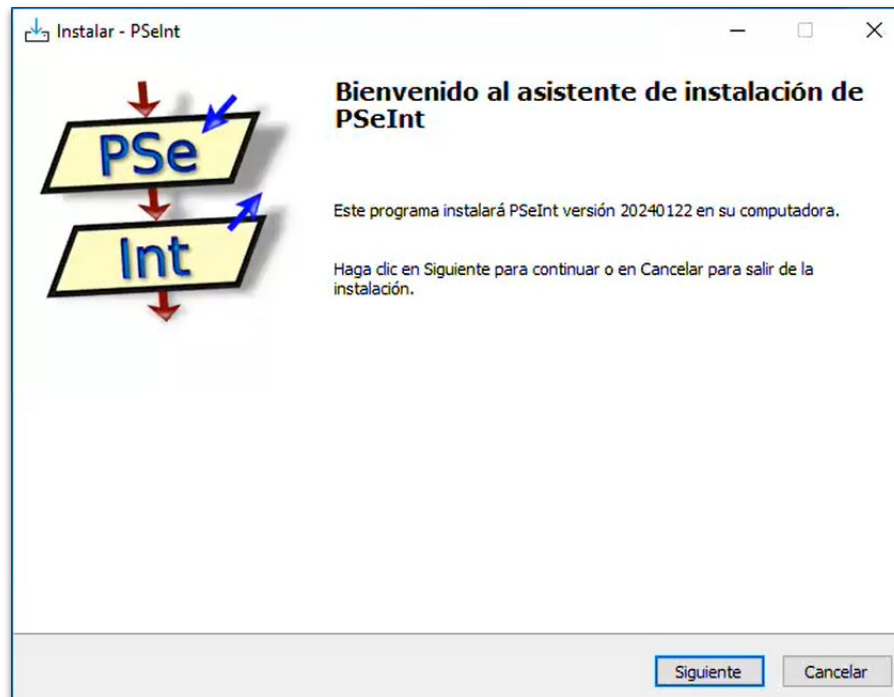
3 - Ejecutá el instalador.

4 - Tendrás que hacer clic en siguiente y luego aceptar los términos y condiciones, luego nuevamente clic en siguiente.

5 - Seleccioná la carpeta de instalación (recomendamos dejar por defecto) y clic en instalar.

5 - Una vez finalizado ya se encuentra listo para ser ejecutado y dando clic en siguiente al finalizar la instalación se abrirá el programa..

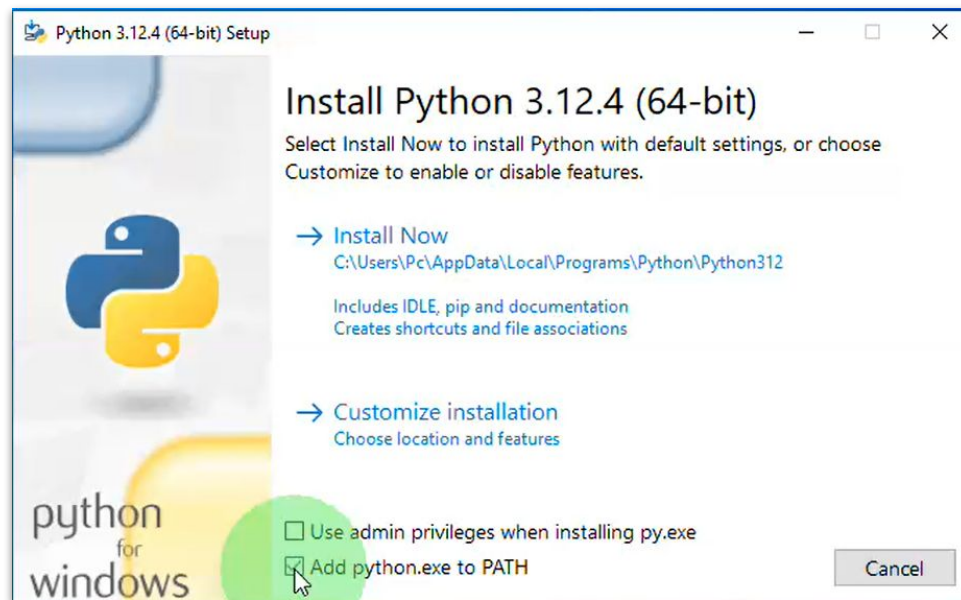
6 - La ventana que aparece cuando inicia el programa solo hay que cerrarla y listo para usar.



Python

Además del video de instalación, te vamos a dejar los pasos para la instalación de Python.

- 1 - Ingresá a <https://www.python.org/downloads/>
- 2 - Se reconocerá el sistema operativo automáticamente, por lo que solo resta descargar la última versión.
- 3 - Una vez descargado, ejecutá el instalador.
- 4 - Seleccioná la opción "Add python.exe Path" (¡sin falta!)
- 5 - Presioná en "Install Now" (quedará la carpeta de instalación por defecto). Si querés cambiar la carpeta de instalación tenés que seleccionar en "Customize installation".
- 6 - Una vez finalizada la instalación, hacé clic en el botón "Close".
- 7 - Python ya está instalado, pero si querés verificar, solo debés buscar en el inicio "Python" y te va a aparecer el programa.



Extensión de Python en VSC

Además del video de instalación te vamos a dejar los pasos para la instalación de la integración de Python con Visual Studio Code:

- 1 - Abri VSC y hacé click en "Extensiones".
- 2 - En la barra de búsqueda, escribí Python y hacé click en el primer resultado (Autor de la extensión: Microsoft).
- 3 - Hacé click en "Install" o "Instalar" y una vez que finalice, VSC te pedirá que reinicies el programa, por lo que se debe reiniciar, y listo, se completó la instalación de la extensión de Python en VSC.





**¡Nos vemos
En la próxima
clase!**

