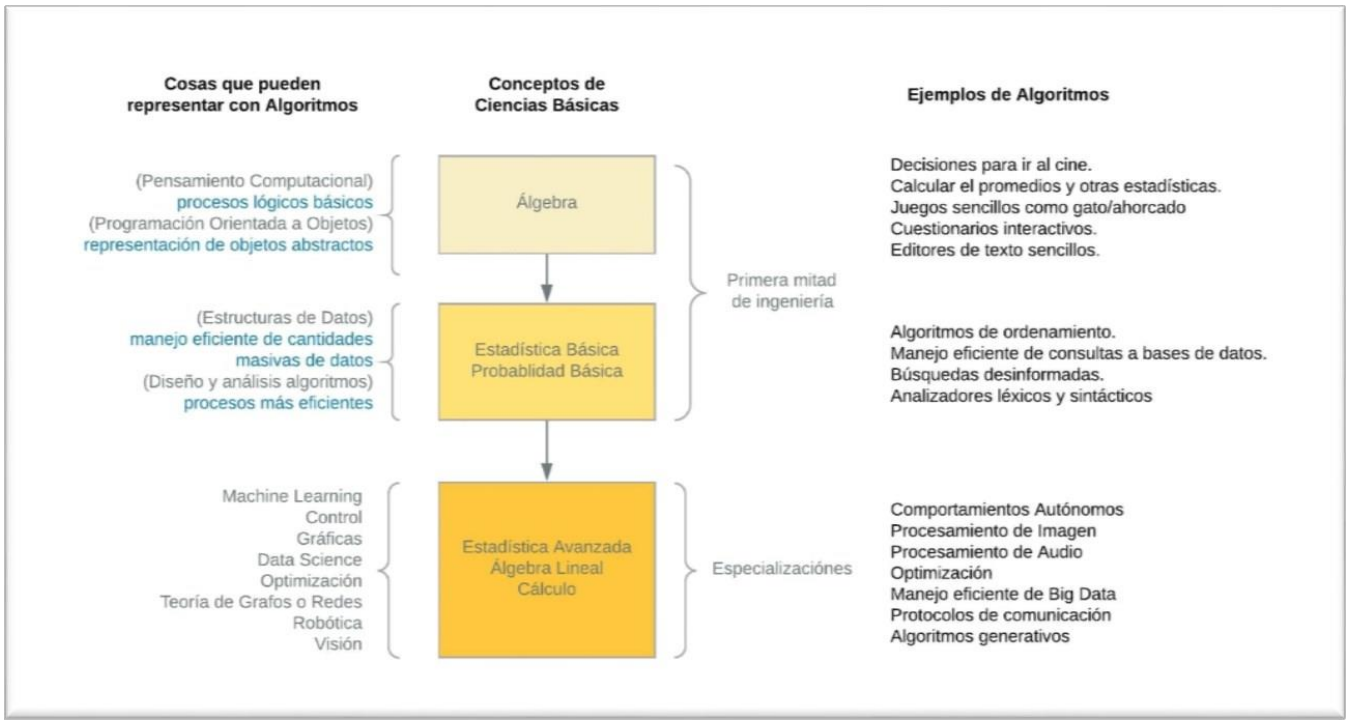
Representación y creación de soluciones

Un poco de contexto:

Los algoritmos representan nuestro conocimiento acumulado sobre cómo resolver problemas de forma automatizada. Saber leer y escribir algoritmos nos abre las puertas a todas las soluciones que hemos creado y documentado hasta la fecha.

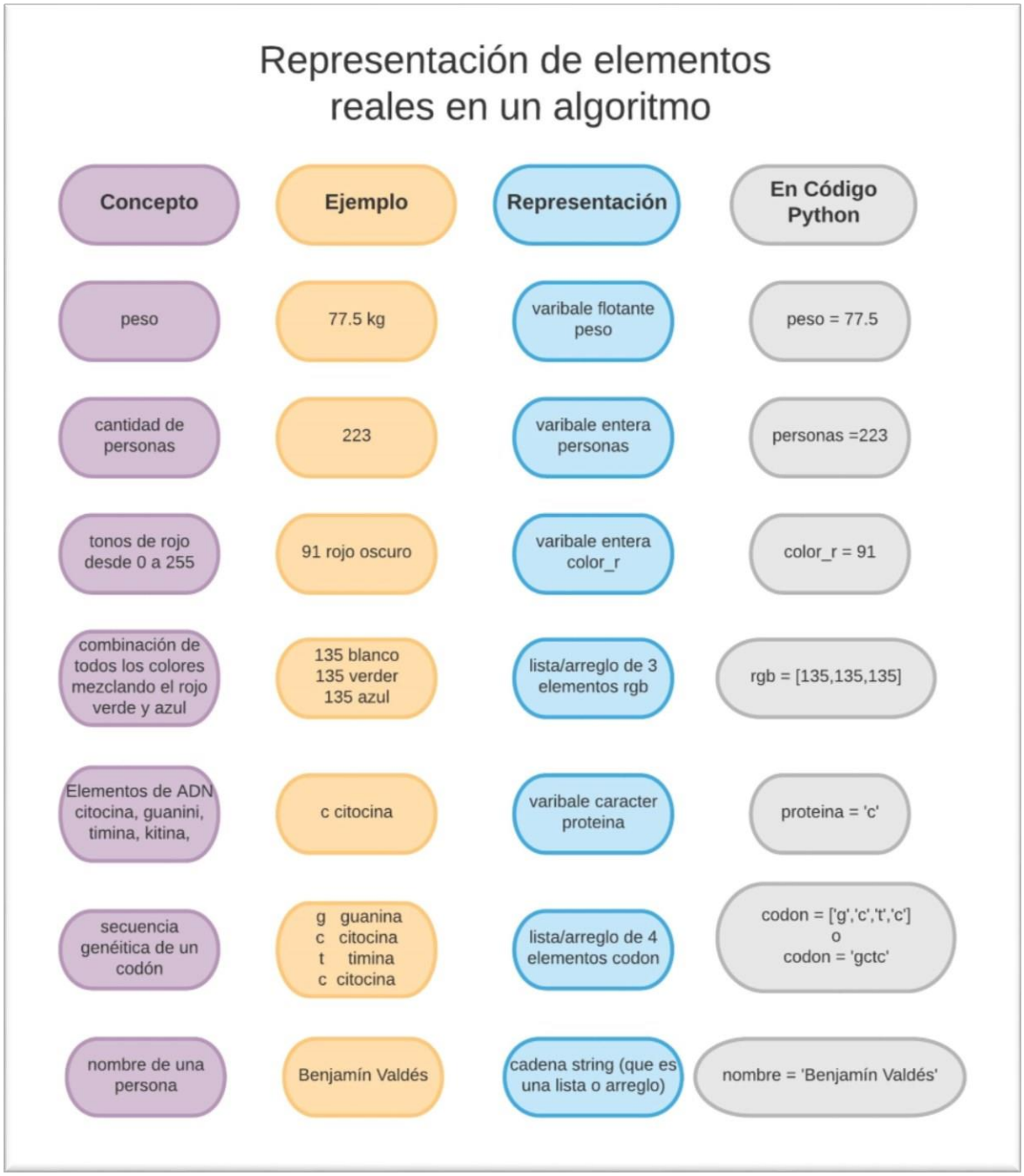
Para poder entender algoritmos más complejos necesitaras cada vez bases más amplias. En el siguiente diagrama se relacionan algunas de las áreas de conocimiento que cursarán en su formación como ingenieros con el alcance que tendrán los algoritmos en estas etapas.



Representación ¿Por o cómo empezar?

Para poder generar soluciones descritas mediante algoritmos es necesario poder representar la realidad de una forma más simple, pero sin perder de vista los puntos que son relevantes para resolver el problema. La diferencia más clara entre los programadores más avanzados no es tanto el lenguaje en que programan, sino lo que pueden representar con él. Por eso entrevistas laborales se les pide resolver problemas y después codificarlos. La representación comienza con el tipo de información que obtenemos del mundo real y como la planteamos en nuestro problema.

En la siguiente figura mostramos ejemplos:



Solucionar Problemas (algoritmos)

Para solucionar cualquier problema, primero debemos:
1) Entender que es lo que nos están pidiendo.
2) Identificar qué información o recursos nos están brindando.

Esto nos llevará a poder ver la situación de manera más clara. Usualmente a este nivel básico, se nos plantean 2 tipos de situaciones:

1) Situación 1: Calcular el área de un círculo.

Crear un algoritmo que calcule el área de un círculo a partir de su radio con la fórmula: área = PI * r^2

Automatizar un proceso que ya se conoce o que está explícito en los requerimientos. Esto es común cuando ya se cuenta con un algoritmo específico o una fórmula o un proceso clara y no ambiguo sobre cómo solucionar el problema. En esta situación, solo es necesario entonces implementar la solución.

2) Situación 2: Encontrar el número más pequeño.

Crear un algoritmo que dados 3 números encuentre el más pequeño.

Automatizar un proceso que no sabemos cómo ocurre, pero del cual tenemos ejemplos de entrada y de salida que esperamos. (De este tipo son los problemas de la programación competitiva)

Casos de prueba

En ambos casos una vez que entendemos el problema generamos ejemplos o casos de prueba. Esto es importante porque nos permite entender dada qué información esperamos qué resultado. Con estos casos probaremos nuestros algoritmos para saber si son correctos o están mal hechos.

Siguiendo los ejemplos anteriores tenemos que:

1. Área de un círculo

Entrada	Salida
3	28.27
5	78.54
-5	78.54

2. Número más pequeño

Entrada	Salida
10, 23, 1	1
200, -23, 9	-23
0, 0, 0	0

Una forma útil de comenzar con el proceso es: ***primero resolver el problema a mano*** con ejemplos concretos, y escribir cada paso de forma detallada.

Si un paso no se puede pasar a una instrucción de programación, entonces el paso ambiguo y debe descomponerse en pasos más específicos. Los algoritmos para estos ejemplos los resolveremos en clase.

Hay varios algoritmos que pueden solucionar el mismo problema. Es importante contemplar que, aunque puede haber varias soluciones, no todas son igual de buenas y es común sacrificar algunos aspectos (velocidad, complejidad, memoria, transparencia, escalabilidad) para mejorar otros. Nosotros como ingenieros tenemos que desarrollar el criterio para saber que conviene más sacrificar en escenarios reales, pero eso ya lo verán cursos posteriores.