



INTRODUCCIÓN A LA MATERIA

Todo lo que debemos conocer antes de empezar

A series of horizontal bars in white, red, and cyan colors located at the bottom of the slide, arranged in a staggered, overlapping fashion.

```
1 01 {
```

```
2  
3  
4  
5 [Bienvenida al curso]
```

- ```
6
7
8 - Presentación
9 - Cronograma de la materia
```

```
10
11
12 }
13
14
```

# Presentación {



¡Hola! Mi nombre es **Stefani Pérez**,  
soy egresada de la **Universidad  
Metropolitana** y seré su profesora  
de Algoritmos y Programación.

[p.stefani16@gmail.com](mailto:p.stefani16@gmail.com)

}

# Cronograma de la materia {



Vamos a hablar sobre lo que veremos  
en el transcurso de estas 3  
semanas:

Cronograma de la materia

}

1  
2 02 {  
3  
4

5 [Ambiente de trabajo]  
6  
7

- 8 - Python
- 9 - Visual Studio Code

10  
11  
12 }  
13  
14

# Python {

**Python** es un **lenguaje de programación de alto nivel** de cuya filosofía se enfoca en la legibilidad del código. Se destaca por ser el más utilizado en áreas como **Inteligencia Empresarial (BI)** e **Inteligencia Artificial (IA)**. Es una herramienta muy poderosa que permite hacer desde páginas web, hasta redes neuronales capaces de todo tipo de actividades.

A pesar de eso, su **sencillez** y **abstracción** de funciones lo hace el lenguaje más utilizado para enseñar programación actualmente.

[Guía de instalación](#)



# Visual Studio Code {

**Visual Studio Code (VSC)** es un **editor de código** desarrollado por Microsoft. Es un **software libre** y **multiplataforma**, y está disponible para Windows, GNU/Linux y macOS.

VSC cuenta con soporte para depuración de código, y dispone de un sinnúmero de extensiones, que básicamente te da la posibilidad de **escribir y ejecutar código en cualquier lenguaje de programación**.

[Guía de instalación](#)



03 {

## [Herramientas a utilizar]

- Google Classroom
- Notion
- Telegram

}



# Google Classroom {



Las entregas se realizarán a través  
de Google Classroom:

**dhjq4u4**

}

## 1 Notion {



3 Tendrán todo el material en **Notion**, podrán visualizarlo  
4 mediante el siguiente link:

5 [Notion de la materia](#)

6 }  
7

## 8 Telegram {



10 Utilizaremos **Telegram** para mantenernos comunicados,  
11 pueden unirse con el siguiente link:

12 [Grupo de telegram](#)  
13  
14 }

# 04 {

## [Introducción a la programación]

- Algoritmos
- Lenguajes de programación
- Función print()
- Python
- Editor de código
- Visual Studio Code

# }

# Algoritmos {

En palabras sencillas un algoritmo es una **serie de instrucciones** o pasos que dado unos datos de entrada, podemos obtener una salida.



# ¿QUÉ ES UN ALGORITMO?

Es la secuencia de pasos que resuelve un problema y es la base de la programación.

Prof. Alexys Lozada



## CARACTERÍSTICAS



### PRECISO

Tiene que resolver el problema sin errores.



### DEFINIDO

Si ejecutas el algoritmo varias veces, los datos de salida serán iguales en cada repetición.



### FINITO

Debe tener un inicio y un final.

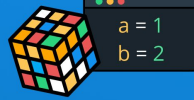
### LEGIBLE

Cualquier persona que vea el algoritmo debe ser capaz de comprenderlo.

## PARTES DE UN ALGORITMO

### ENTRADA

Son los datos que se le dan al algoritmo.



$a = 1$   
 $b = 2$

### PROCESO

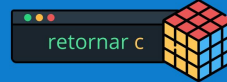
Operaciones que se hacen con los datos.



$c = a + b$

### SALIDA

Resultado final que se obtiene de las operaciones, en este caso será 3.



retornar c

Aprende como funcionan los algoritmos en:

[ed.team/cursos/algoritmos](https://ed.team/cursos/algoritmos)



# Lenguajes de programación {

Es un **sistema de notación** para escribir algoritmos, es la herramienta que utiliza el programador para comunicarse con la computadora, indicando que es lo que tiene que hacer.



# Tipos de lenguajes {

## Alto nivel



Lenguajes **diseñados para poder ser entendidos por humanos** y que nos permiten abstraer instrucciones y/o datos.

*Ejemplo: Python, Java, JavaScript*

## Bajo nivel



Lenguajes que **se acercan más al código máquina** (1100101) y que son muy difíciles de interpretar por los humanos.

*Ejemplo: Ensamblador*

}

# Función print() {

En Python, print es una función incorporada que se utiliza para **mostrar información en la consola** o en la salida estándar del sistema.

La función print toma uno o más argumentos separados por coma y los muestra en la pantalla.



```
print("Soy un mensaje que aparecera en la consola")
```

}






**Hello world!**


# Tipos de lenguajes {

## Python



```
print("Hello world!")
```

## Assembler



```
.data

mensaje: .asciiz "Hello world!"

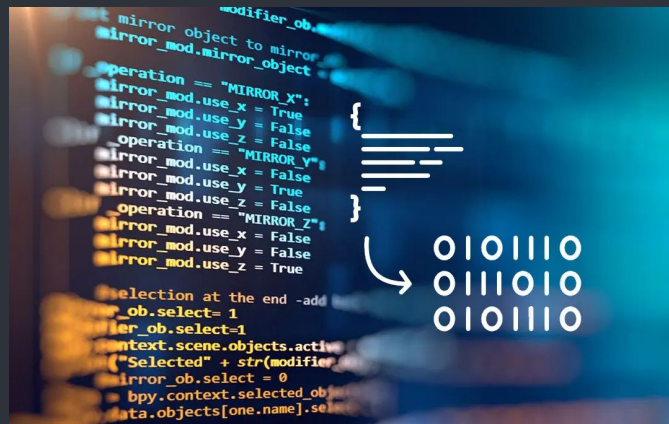
.text

li $v0 4
la $a0 mensaje

syscall
```

# Funcionamiento de los lenguajes {

Los algoritmos escritos en lenguajes de programación deben ser **traducidos a lenguaje máquina**, esto se hace mediante un programa auxiliar. Existen dos tipos de programa para esta traducción: **compiladores** e **interpretadores**.



# Tipos de programas traductores {

## Compilador

Programa que **traduce el código fuente completo** de un programa en lenguaje de programación a un archivo ejecutable en lenguaje de máquina. El compilador realiza esta tarea en un solo paso.

## Interpretador

Programa que **lee y ejecuta el código fuente** de un programa **línea por línea**. En lugar de compilar todo el programa de una sola vez, el interpretador lo lee y ejecuta en tiempo real.

}

# Python {

Python es un lenguaje de programación **interpretado**, muy utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML).

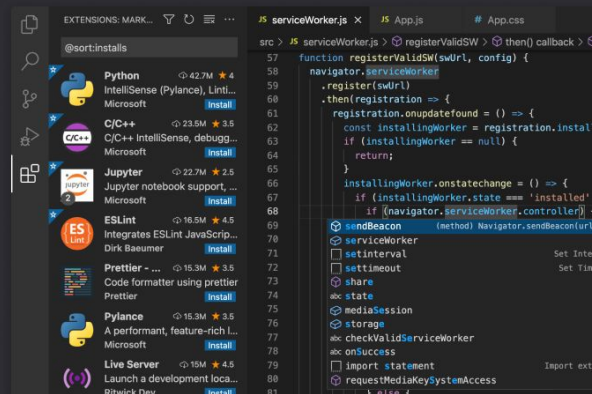
Es uno de los lenguajes más usados hoy en día y se caracteriza por ser **fácil de aprender**.



}

# Editor de código {

Un editor de código es una herramienta que **permite editar archivos que contienen texto** (como los archivos donde programamos) y que en consecuencia resultan ser muy útiles para programar.



# Visual Studio Code {

Visual Studio Code (VSC) es un **editor de código**, que cuenta con soporte para depuración de código, y dispone de varias extensiones. Da la posibilidad de **escribir y ejecutar código** en cualquier lenguaje de programación.



1           05 {  
2  
3

4  
5           [Datos]  
6  
7

- 8           - ¿Qué es un dato?  
9           - Tipos de datos  
10

11           }  
12  
13  
14



# ¿Qué es un dato? {

Un dato es una **pieza de información** que describe una característica de alguna entidad u objeto.

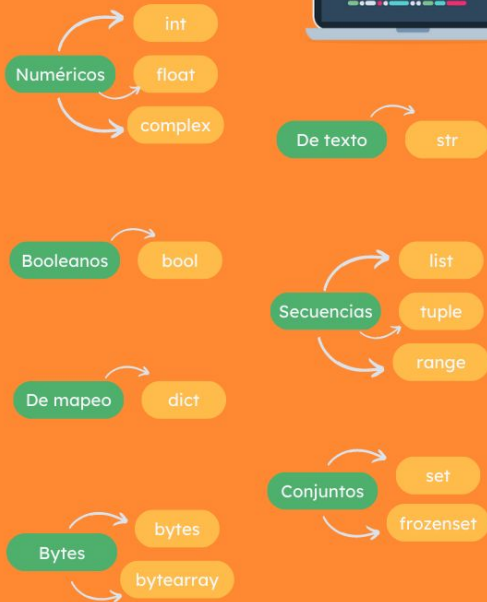
*Ejemplo: La edad de una persona, el nombre de un lugar...*

Se subdividen en **tipos de datos**. Aquellos datos que son del mismo tipo (o de tipos relacionados) comparten características. Esto ultimo, nos permite hacer operaciones con estos datos.

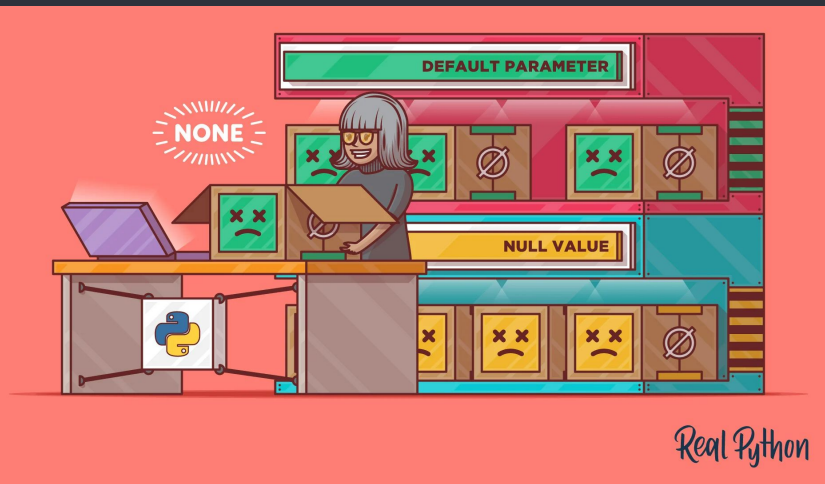
*Ejemplo: Ordenar una lista de nombres alfabeticamente, sumar dos cantidades numéricas...*

}

# Tipos de datos en Python



Aunque **None** no es considerado un tipo de dato en Python, es un objeto que representa la **ausencia de un valor** y puede ser asignado a variables y utilizado en expresiones como cualquier otro valor.



Real Python

06 {

## [Variables]

- ¿Qué es una variable?
- Asignación de variables
- Convención para nombrar variables
- Palabras reservadas
- Función type()
- Numeros, Booleanos, Strings
- Casteos

}

# ¿Qué es una variable? {

Es un **espacio en la memoria** de la computadora que se utiliza para almacenar un valor o una referencia a un valor. Las variables permiten guardar y manipular datos en el programa.

Las variables pueden contener diferentes tipos de datos, como números, texto, valores booleanos, listas, diccionarios, entre otros. Para utilizar una variable en un programa, se debe declarar su nombre, su tipo de dato (*no es necesario en Python*), y luego asignarle un valor a esa variable utilizando el **operador de asignación =**.



```
nombre_variable = "¡Hola! Soy una variable"
```

# Asignación de variables {

La asignación es el proceso mediante el cual se le indica a Python el valor que deberá guardar en la variable mediante el uso del **operador de asignación =**.

**x = y** —  — **y = x**

}

# Asignación de variables {

Las variables **deben ser asignadas** antes de poder ser usadas.

```
x = 1
y = q
x = 2
print(y)
```

```
>>> NameError: name
'q' is not defined
```

```
x = 1
q = 2
y = x / q
print(y)
```

```
>>> 0.5
```

# Convención para nombrar variables {

01

Debe ser **descriptivo y relevante** para el valor que almacena.

02

Debe comenzar con una **letra minúscula**.

03

Se pueden separar las palabras utilizando **guiones bajos \_**

04

Evitar nombres de variables que sean iguales a **palabras reservadas** de Python

05

Combinar palabras en una sola cadena

}

# Palabras reservadas {

las palabras reservadas son palabras que tienen un significado especial y se utilizan para definir la estructura y la sintaxis del lenguaje de programación. Estas palabras tienen un uso específico en el lenguaje y **no se pueden utilizar como nombres de variables, funciones o clases.**



}



# Palabras reservadas {

|        |          |         |          |        |
|--------|----------|---------|----------|--------|
| False  | class    | finally | is       | return |
| None   | continue | for     | lambda   | try    |
| True   | def      | from    | nonlocal | while  |
| and    | del      | global  | not      | with   |
| as     | elif     | if      | or       | yield  |
| assert | else     | import  | pass     |        |
| break  | except   | in      | raise    |        |

}

# Función type() {

la función type() se utiliza para **obtener el tipo de dato de un objeto** en particular. La función type() toma como argumento un objeto y retorna el tipo de dato correspondiente a ese objeto.

Por ejemplo, se puede utilizar la función type() para obtener el tipo de dato de una variable:



```
x = 5
print(type(x)) # Imprime <class 'int'>
```

}

# Números {

En Python podemos manejar tres tipos de números (**int**, **floats**, **complex**) aunque comúnmente estaremos manejando los **int** y **floats**.

## Enteros (int)



```
a = 10
```

```
b = -5
```

## Flotantes (float)



```
c = 3.14
```

```
d = -2.5
```

## Complejos (complex)



```
e = 3 + 2j
```

```
f = -1j
```

}

# Números decimales no exactos {

En Python (y en muchos otros lenguajes de programación) los **números decimales (flotantes) no son exactos** debido a la forma en que se almacenan y se representan en la memoria de la computadora.

Para minimizar el impacto de estas imprecisiones, se pueden utilizar módulos de Python como **decimal** o **numpy** que permiten trabajar con números decimales con mayor precisión.

```
print(0.2 + 0.1)
```

```
0.30000000000000004
```

```
print(1.13 - 1.1)
```

```
0.02999999999999998
```

# Booleanos {

En Python, los booleanos son un tipo de dato que representa dos valores:

**Verdadero (True)** o **Falso (False)**

Los booleanos se utilizan en Python para realizar **operaciones lógicas y de comparación**, cómo comprobar si una condición es verdadera o falsa.



```
es_programador = True
```

```
es_abogado = False
```

# Strings {

En Python, un string es una **secuencia de caracteres** que se utiliza para representar texto. Los strings se pueden definir utilizando **comillas simples** (') o **dobles** (").



```
mensaje = "¡Hola mundo!"
mensaje_2 = 'Hello world!'
```

# Concatenación {

La concatenación es la **operación de unir dos o más strings en uno solo**. En Python, se puede utilizar el **operador +** para concatenar strings:



```
saludo = "Hola"
nombre = "Stefani"
mensaje = saludo + " " + nombre + "!"
print(mensaje) # Hola Stefani!
```

# Multiplicar String {

En Python se puede multiplicar un string utilizando el **operador** \*. Al multiplicar un string por un número entero, se crea una nueva cadena que consiste en la concatenación del string original ese número de veces.



```
patron = "*-*"
nuevo_patron = patron * 5
print(nuevo_patron) # Imprime "*-*-*-*"
```



# Función len() {

len() es una función que se utiliza para obtener la **longitud de una secuencia**, incluyendo **strings**. La función len() devuelve el número de elementos de la secuencia, es decir, el **número de caracteres** en el caso de los strings.




```
tamano_cadena = len("Hola mundo!")
print(tamano_cadena) # Imprime 11
```

}

# Metodo.format() {

En Python, el método .format() es una forma de **formatear cadenas de texto** que permite agregar valores de variables o expresiones dentro de una cadena de texto de manera dinámica.

Los argumentos de la función .format() se colocan dentro de las llaves en el orden en que se desean mostrar.



```
numero_1 = 15
numero_2 = 12334
cadena_formateada = "Los numeros guardados en el programa son {} y {}".format(numero_1, numero_2)
Los numeros guardados en el programa son 15 y 12334
```

}

# Metodo.format() {


También es posible utilizar una **sintaxis abreviada** utilizando la letra f al comienzo de una cadena, lo que se conoce como "f-strings" o "formatted string literals" en inglés. Las f-strings permiten incrustar valores de variables o expresiones dentro de una cadena de texto de manera similar a la función .format(), pero con una sintaxis más concisa y legible.

```
numero_1 = 15
numero_2 = 12334
cadena_formateada = f"Los numeros guardados en el programa son {numero_1} y {numero_2}"
Los numeros guardados en el programa son 15 y 12334
```


# Casteos {

El término casteo (o "casting") se refiere a la **conversión explícita de un objeto de un tipo de datos a otro tipo de datos**.


Python es un lenguaje de programación de tipado dinámico, lo que significa que el tipo de datos de una variable puede cambiar dinámicamente durante la ejecución del programa.



```
numero = 1.5
numero = int(numero)
print(numero) #1
```



```
cadena = "30"
numero = int(cadena)
print(numero) #30
```




```
numero_decimal = 3.7
cadena = str(numero_decimal)
print(cadena) #"3.7"
```

}

# Casteos {

Para concatenar otros tipos de datos en strings, se deben transformar primero estos a datos de **tipo cadena**.



```
resultado = 10
respuesta = "El resultado de la operación es " + str(resultado)
print(respuesta)
El resultado de la operación es 10
```

}

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

07 {

## [Operadores aritméticos]

- Operadores aritméticos
- Función input()

}

# Operadores aritméticos {

[ + ]

Suma

```
a = 5
b = 3
c = a + b
print(c) # Imprime 8
```

[ - ]

Resta

```
d = 9
e = 4
f = d - e
print(f) # Imprime 5
```

[ \* ]

Multiplicación

```
g = 2
h = 3
i = g * h
print(i) # Imprime 6
```

[ / ]

División

```
j = 10
k = 3
l = j / k
print(l) # Imprime 3.3333333333333335
```

}

# Operadores aritméticos {

## División entera

[ // ]

```
s = 10
t = 3
u = s // t
print(u) # Imprime 3
```

## Potenciación

[ \*\* ]

```
p = 2
q = 3
r = p ** q
print(r) # Imprime 8
```

## Módulo

[ % ]

```
m = 10
n = 3
o = m % n
print(o) # Imprime 1
```

Es un operador matemático que **retorna el resto**, de la división entre dos números. Si la división es exacta, entonces retorna 0.



# Función input() {

Es una función que **recibe un dato a traves de la consola**, el dato recibido siempre será de **tipo cadena**, opcionalmente se le puede pasar un string como mensaje de solicitud de input.

```
name = input("Ingrese su nombre: ")
print(name)
NOMBRE INGRESADO
```

```
numero = int(input("Ingrese un numero: "))
print(numero)
NUMERO INGRESADO
```

}

# Ejercicio 1 {

Se debe realizar un programa que le pida al usuario su nombre, su apellido y su carrera utilizando “**inputs**”. Luego, se debe mostrar estos datos por consola con el formato de su preferencia.

}

## Ejercicio 2 {

Pedir al usuario dos números y después mostrar por consola los resultados de la **suma**, la **resta**, la **multiplicación**, la **division**, la **división entera**, el módulo y la **potenciación** de los dos números ingresados.

}

## Tarea

### Ejercicio 3 {

Cree un programa que pida al usuario que ingrese su **nombre** y su **edad**. Imprima un mensaje dirigido a ellos que les indique el año en que cumplirán/cumplieron 18 años.

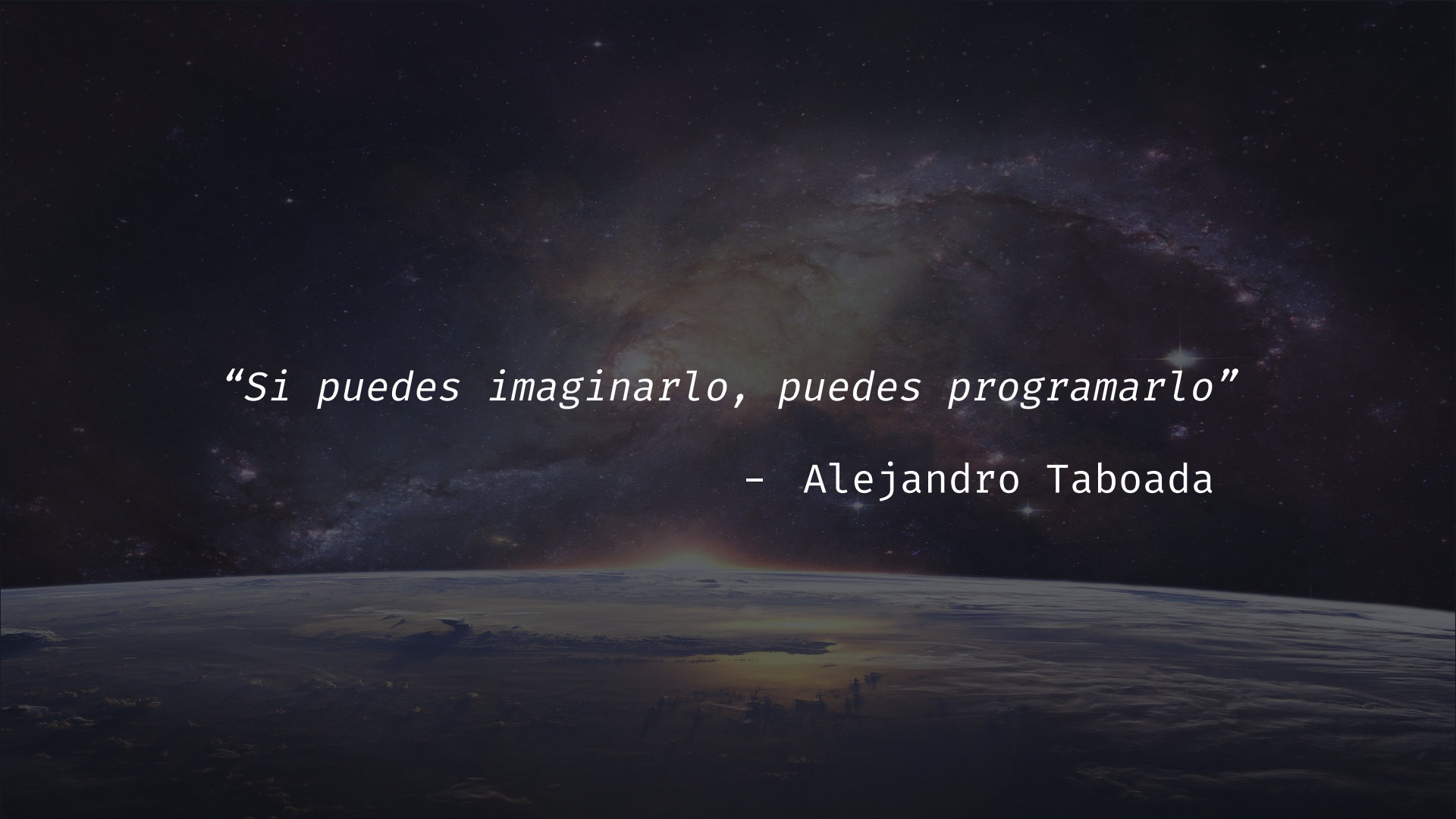
}

## Tarea

### Ejercicio 4 {

Pedir al usuario el valor de un producto y la cantidad que se va a comprar de dicho producto, después con estos datos se debe calcular el **subtotal de la compra** (precio sin iva), el **iva** (asumir iva de 16%) y el **total a pagar** (subtotal + iva).

}



*“Si puedes imaginarlo, puedes programarlo”*

- Alejandro Taboada