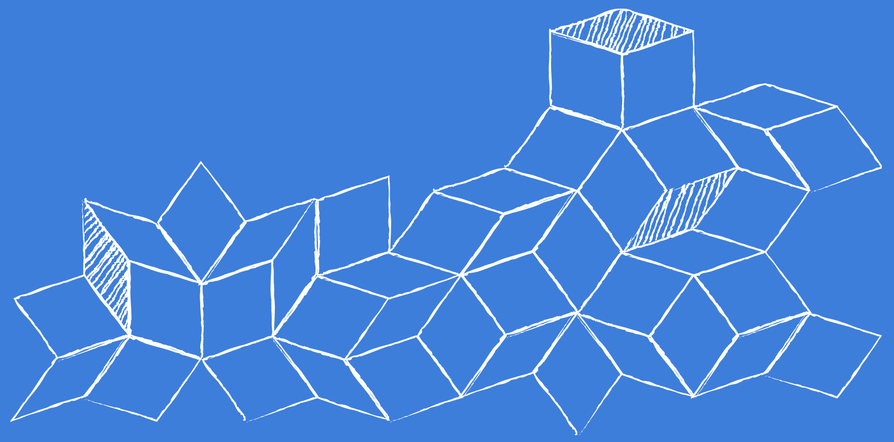
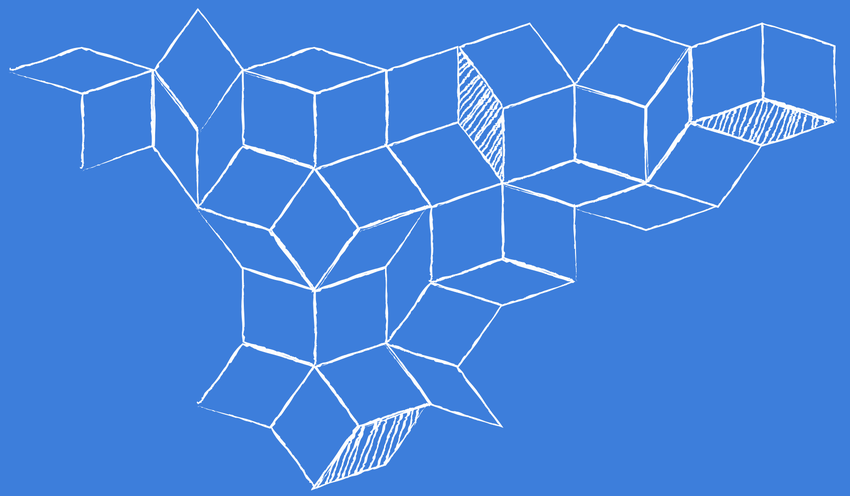


STEMUC

Academia para **escolares**



Suma MAT UC

preingeniería

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

PRIMEROS PASOS
EN PROGRAMACIÓN

Clase 1

Algoritmos, datos y operadores

¿Cuál es el objetivo de este curso?

Aprender a programar

¿Cómo empezar?

Primero, debemos entender
el **concepto de algoritmo**:

- Algoritmo es una secuencia de pasos.
- Un programa está **escrito en un lenguaje**, e **implementa un algoritmo**.

¿Cómo empezar?

¿Cómo programar?
Usando una herramienta apropiada, un computador.

¿Cómo hago que funcione?
Le decimo qué queremos hacer con los datos, es decir, especificándole un algoritmo.

¿Cómo le decimos qué hacer?
Hablando su lenguaje, el lenguaje binario.

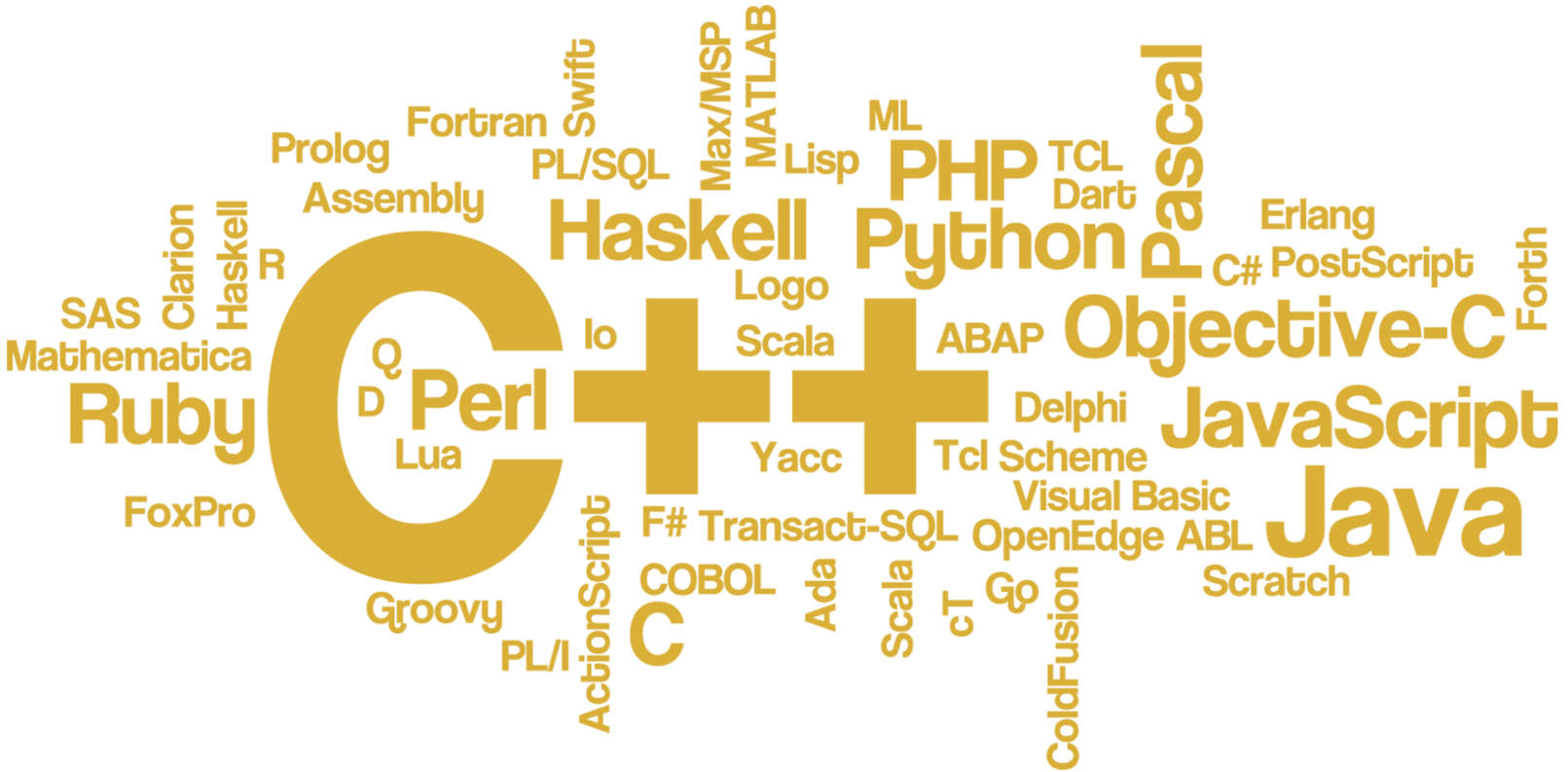
¿Cómo empezar?

¿Qué es un lenguaje de programación?
Es un lenguaje comprensible por le humano (programador) y el computador.

> *Los lenguajes de programación definen y compilan un conjunto de instrucciones para la CPU (Unidad central de procesamiento) para realizar cualquier tarea específica. Cada lenguaje de programación tiene un conjunto de palabras clave junto con la sintaxis que utiliza para crear instrucciones.*

¿Cómo empezar?

¿Qué lenguajes de programación existen?



¿Cómo empezar?

¿En qué lenguaje programaremos?

python™

¿Cómo empezar?

¿Por qué Python?

C++

```cpp
#include <iostream>
int main () {
    std::cout << "Hello World !" << std::endl;
    std::cin.get();
    return 0;
}
```

¿Cómo empezar?

¿Por qué Python?

Java

```java
public class HelloWorld {
  public static void main (String[] args) {
    System.out.println("Hello World !");
  }
}
```

¿Cómo empezar?

¿Por qué Python?

Python

```python
print("Hello Wordl!")
```

Con lo anterior, ya más claro…

Empecemos a hablar de los algoritmos

Algoritmos

Algoritmo (sust.)

Palabra utilizada por programadores cuando…
no quieren explicar lo que hicieron.

Algoritmos

RAE:

Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema.

Wikipedia:

Un algoritmo es cualquier cosa que funcione paso a paso, donde cada paso se puede describir sin ambigüedad.

Algoritmos

¿Qué se necesita?

- Pensamiento crítico (analítico).
- Modelar (imaginar) y descomponer un problema.
  - Descomposición y abstracción
- Prever una secuencia de pasos para resolver el problema
  - Crear algoritmos

Tips

- Un buen entendimiento o conceptualización de un problema ayuda a resolverlo mejor.
- Un buen algoritmo se programa o implementa mejor.

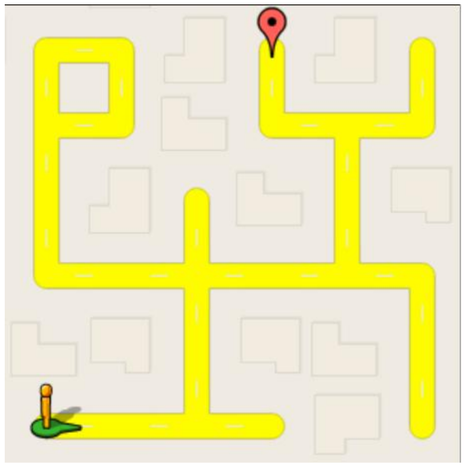
Algoritmos

Pensamiento Computacional

1. Descomponer
   - Dividir el problema en partes más pequeñas y manejables.
2. Reconocer patrones
   - Buscar similitudes entre las distintas partes y problemas que ya han solucionado.
3. Abstraerse
   - Concentrarse en lo importante e ignorar lo irrelevante.
4. Algoritmo
   - Diseñar pasos o reglas que solucionen cada parte del problema.

Algoritmos

¿Qué hace el siguiente algoritmo?



Entendemos qué son los algoritmos, pero

¿Cómo se programa?

¿Cómo se programa?

Tanto en Python como en todos los lenguajes de programación, existen distintos tipos de datos que podemos manipular.

En esencia, programamos usando texto que se interpreta como código.

¿Cómo se programa?

```
entrada = 0
suma = 0
contador = 0

while contador < 10:
    entrada = int(input("Ingrese un número positivo: "))
    if (entrada > 0):
        # Solo sumamos positivos
        suma = suma + entrada
        contador = contador + 1

print("La suma es: " + str(suma))
```

Expresiones

Hay algunas variables:

- `entrada` Para **guardar el dato** ingresado por el usuario. Por ejemplo: el número 0.
- `suma` Para **guardar la suma** de los números ingresados por el usuario. Parte en 0.
- `contador` Para **guardar la cantidad** de números ingresados por el usuario.

Expresiones

También hay instrucciones:

- `while` Permite ejecutar código, **mientras** se cumpla cierta condición.
- `if` Determina **si** se cumplen ciertas condiciones
- `+` **Suma** valores.
- `>` **Compara** valores.
- `print()` **Escribe** el valor que se le entregue.
- `input()` **Lee** el texto que le entregue el usuario.

Expresiones

Todos los programas que escribiremos se pueden construir con:

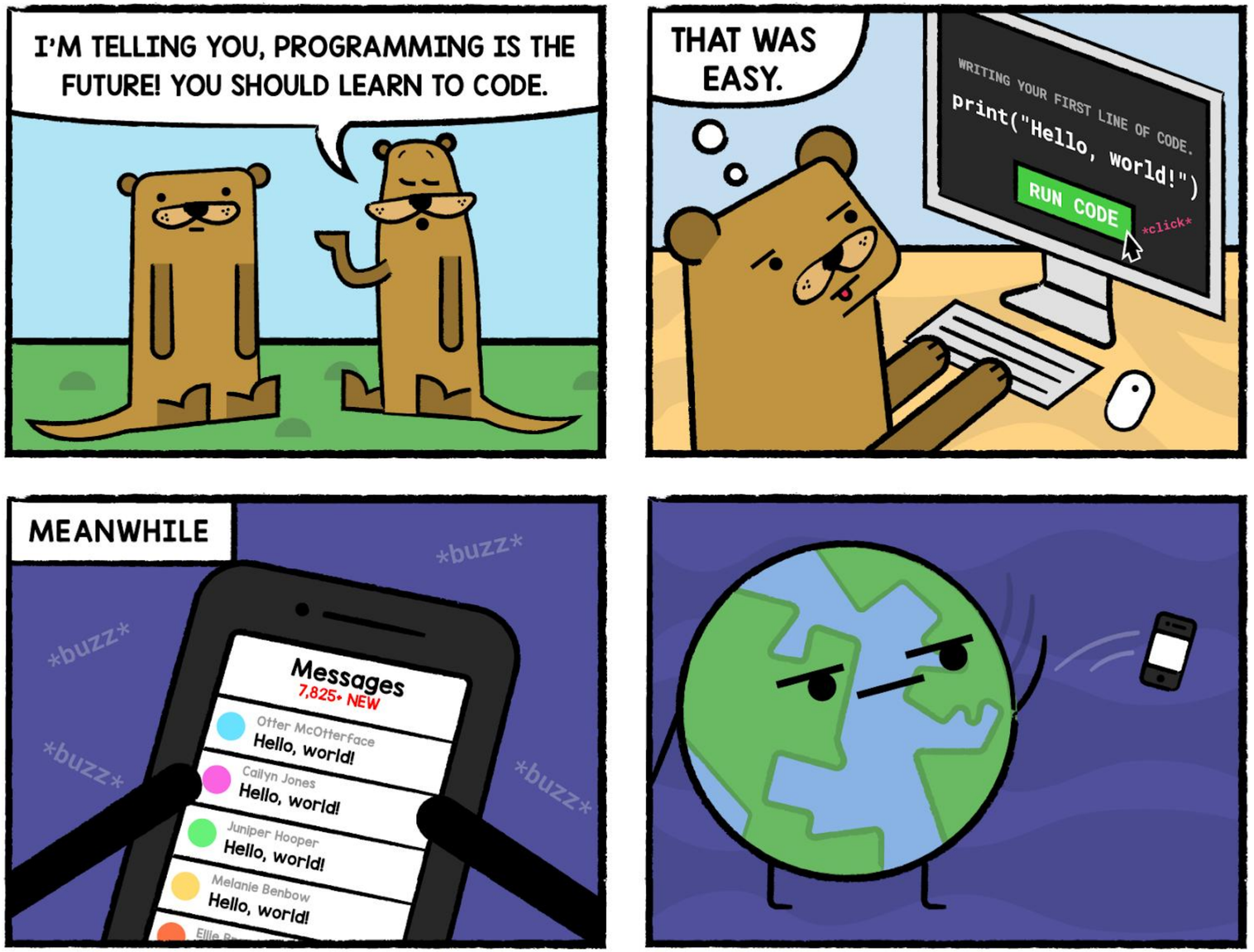
- Datos de **entrada** (*input*) que se leen.
- **Variables** que recuerdan datos.
- Instrucciones **condicionales** que se ejecutan dependiendo de una condición.
- Datos de **salida** (*output*) que se escriben.
- **Operaciones** que calculan datos.
- Instrucciones **repetitivas** que se ejecutan múltiples veces dependiendo de una condición.

Ahora que conocen las bases

Empecemos a programar un poco

OTTER THIS WORLD

@REIKACANJA



When your code outputs "Hello World!"

Empecemos

Comenzaremos aprendiendo instrucciones básicas de Python como **`print`**, y tipos de datos básicos como **`int`**, **`float`**, **`str`** y **`bool`**.

print

Comenzamos con la instrucción `print`. Esto será esencial para nosotros, pues nos permitirá escribir en la salida estándar, que generalmente es la consola o terminal.

```
# Primero notemos que es posible escribir
# comentarios anteponiendo hashtag.
print("Hola")
print("Hola de nuevo")
print('Intenten imprimir algo ustedes')
```

print

Veamos un detalle de la instrucción `print()`:

```
print("Es", "fácil", "multiplicar", "por", 0)
print("Es" + "fácil" + "multiplicar" + "por" + str(0))
```

Lo anterior mostrará en consola lo siguiente:

```
Es fácil multiplicar por 0
Esfácilmultiplicarpor0
```

input

La instrucción `input`, nos permitirá leer información que nos entregue el usuario a través en la salida estándar.

```
nombre = input("Nombre: ")
edad = input("Edad: ")
```

Nota: La instrucción `input`, siempre nos entregará un texto, independientemente de que parezca un número u otro tipo de dato.

Tipos de datos

Números

- *int* 4
- *float* 4.0

Texto

- *str*

```
"Texto comillas dobles"
'Texto comillas simples'
```

Booleano

- *bool* True
  False

Tipos de datos

```
5                   # int
5.0                 # float

"Hello world"       # string
'Hello world'       # string

True                # bool
False               # otro bool

true                # nada por sí solo
```

Tipos de datos

En caso de dudas, podemos **verificar de qué tipo es** un dato utilizando la función *type*( ). Esta retorna el tipo de dato, y podemos visualizarlo imprimiéndolo.

```
print(type(5))              # verificar que es un int
print(type(5.0))            # verificar que es un float
print(type("texto"))        # verificar que es un string
print(type(True))           # verificar que es un bool
```

Tipos de datos

También podemos **convertir datos** de un tipo a otro utilizando las funciones `int()`, `float()`, `str()`.

A esto se le denomina ***casting*** o castear, informalmente en español.

```
int("5")     # transforma de string a int
float(5)     # transforma de entero a float
str(543)     # transforma de int a string
```

```
5
5.0
'543'
```

Operadores

En todos los lenguajes de programación, podemos utilizar operadores para realizar operaciones sobre valores o variables.

En Python, estos operadores se dividen en los siguientes grupos principales:

- Operadores de asignación
- Operadores comparativos
- Operadores aritméticos
- Operadores lógicos

Hay más, pero no los veremos en este curso.

Operadores aritméticos, sobre números

Identidad

```
+2              # Identidad +2 = 2
+0              # Identidad +0 = 0
```

Negación

```
-2              # Negación -2 = -2
-7 -5           # Negación -7 -5 = -12
```

Suma

```
4 + 2           # Suma 4+2 = 6
5 + (4 + 3)     # Suma 5+(4+3) = 12
```

Resta

```
4 - 2           # Resta 4-2 = 2
5 - (8 - 2)     # Resta 5-(8-2) = -1
```

Multiplicación

```
3 * 4           # Multiplicación 3*4 = 12
```

Exponenciación

```
2 ** 3          # Exponente 2**3 = 8
```

Operadores aritméticos, sobre números

División

```
5 / 2          # División 5/2 = 2.5
3.5 / 2        # División 3.5/2 = 1.75
```

División entera

```
5 // 2         # Resta 5//2 = 2.0
3.5 // 2       # Resta 3.5//2 = 1.0
```

Módulo o resto

```
6 % 4          # Módulo 6%4 = 2
7 % 2          # Módulo 7%2 = 1
1 % 5          # Módulo 1%5 = 1
```

Operadores aritméticos, sobre números

Tabla de resumen

Operador	Descripción	Ejemplo	Precedencia
**	Exponente	2 ** 3	1
+	Identidad	+2	2
-	Negación	-5	2
*	Multiplicación	5 * 3	3
/	División	7 / 2	3
//	División entera	7 // 2	3
%	Módulo	5 % 4	3
+	Suma	3 + 1	4
-	Resta	4 - 2	4

Nota: Para evitar problemas con la precedencia, usen paréntesis.

Operadores aritméticos, sobre textos

Concatenar (con el operador +)

```
"Yo soy" + "tu padre"     # "Yo soytu padre "
"Yo soy" + " tu padre"    # "Yo soy tu padre "
```

Repetición (con el operador *)

```
"Ja" * 5                  # "JaJaJaJaJa"
```

Mezclas entre distintas operaciones

```
"N" + "o" * 9             # " Nooooooooo"
("N" + "o") * 9           # " NoNoNoNoNoNoNoNoNo"
```

Operadores comparativos

Los operadores de comparación se usan para comparar dos valores o variables. Siempre retornan un booleano: `True` o `False`.

```
4 == 4    # Igual:         Es True ssi 4 es igual a 4
5 != 4    # Distinto:      Es True ssi 5 es desigual a 4
3 > 2     # Mayor:         Es True ssi 3 es mayor a 2
3 >= 2    # Mayor igual:   Es True ssi 3 es mayor o igual a 2
2 < 1     # Menor:         Es True ssi 2 es menor a 1
2 <= 1    # Menor igual:   Es True ssi 2 es menor o igual a 1
```

También podemos aplicarlos a *strings*.

```
"Hola" == "Hola"                       # True
"No puede ser" != "No puede zer"       # True
```

Operadores lógicos

Y (and)

```
(2 > 1) and (3 < 7)    # True  and True  = True
(2 > 1) and (3 < 2)    # True  and False = False
(0 > 1) and (3 < 2)    # False and False = False
```

O (or)

```
(2 > 1) or (3 < 7)     # True  or True  = True
(2 > 1) or (3 < 2)     # True  or False = True
(0 > 1) or (3 < 2)     # False or False = False
```

Negación (not)

```
not (2 > 1)            # Not True  = False
not (3 < 2)            # Not False = True
```

Hoy vimos muchas cosas

¿Qué es lo que aprendieron hoy?

Ve a

www.wooclap.com

Introduce el código

ASRUFH