

Clase 1. Introducción



TRABAJADORES
INFORMÁTICOS

organizados en AGC



Ministerio de Trabajo,
Empleo y Seguridad Social
Argentina

¿Que es un
algoritmo?

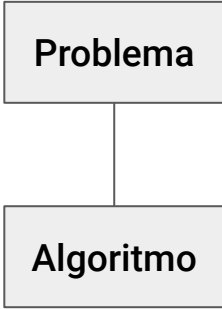
Algoritmo

¿Que es un algoritmo?

Un **algoritmo** es una secuencia detallada de pasos que nos permite resolver un **problema**

Problema

Algoritmo



```
graph TD; A[Problema] --- B[Algoritmo]
```

The diagram consists of two light gray rectangular boxes. The top box is labeled 'Problema' and the bottom box is labeled 'Algoritmo'. A thin vertical line connects the bottom center of the 'Problema' box to the top center of the 'Algoritmo' box, indicating a direct relationship or flow from the problem to the algorithm.

¿Que es un programa?

Un **algoritmo** es una secuencia detallada de pasos que nos permite resolver un **problema**

Problema

Algoritmo

Programa

¿Que es un programa?

Un **algoritmo** es una secuencia detallada de pasos que nos permite resolver un **problema**

Problema

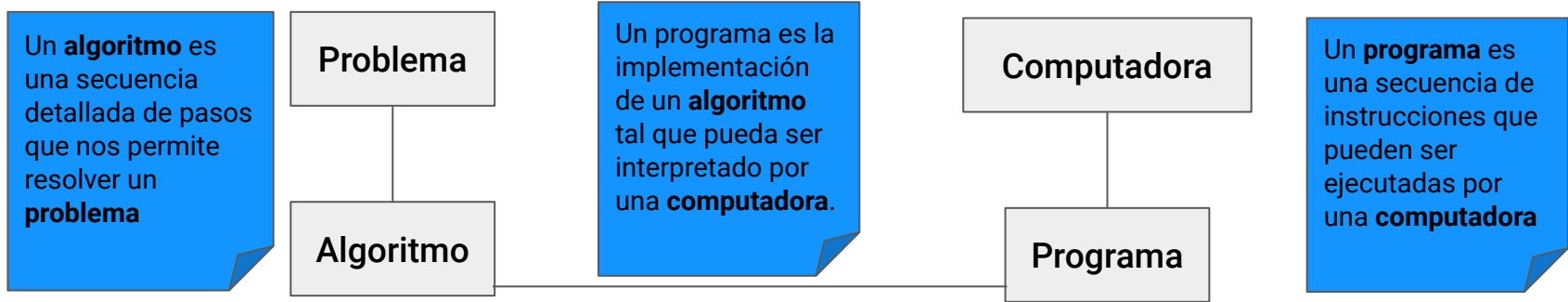
Algoritmo

Computadora

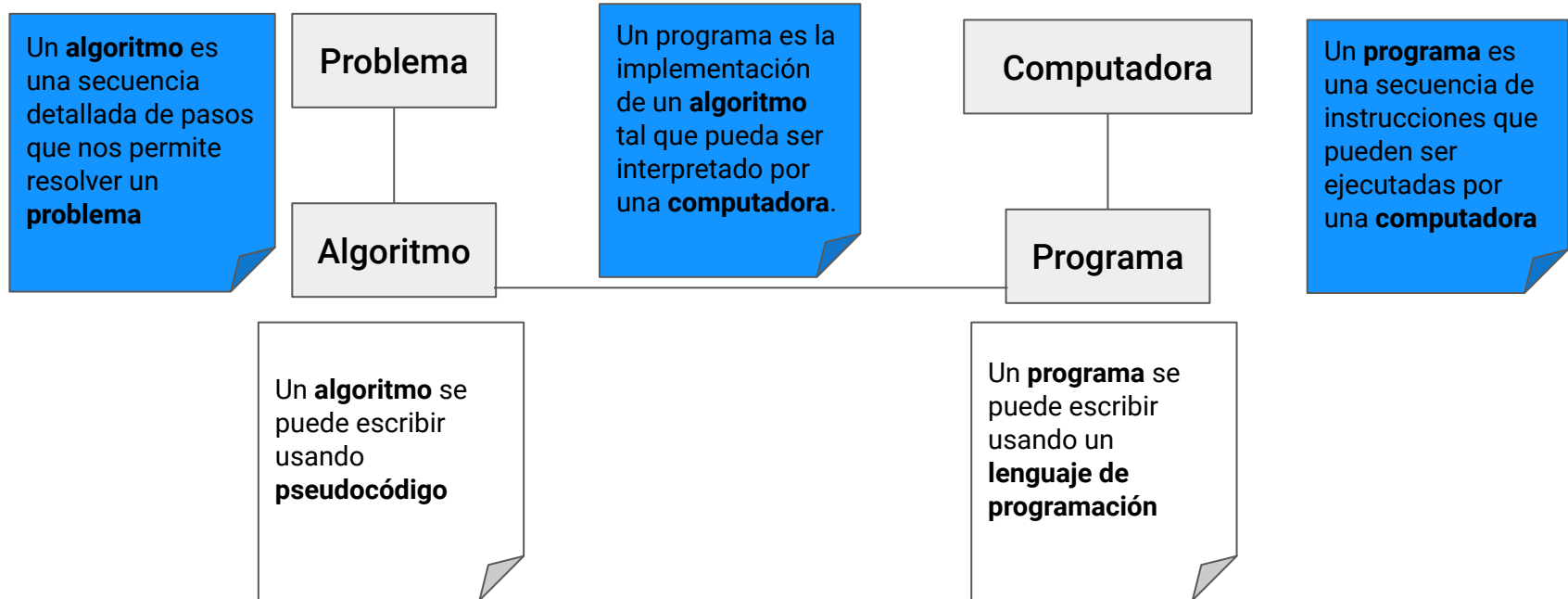
Programa

Un **programa** es una secuencia de instrucciones que pueden ser ejecutadas por una **computadora**

¿Cómo se relaciona **programa** y un **algoritmo**?



¿Cómo se relaciona programa y un algoritmo?



Python

Python es un lenguaje de programación que vamos a usar en este curso y en otros. Nos permite crear programas a partir de algoritmos que la computadora va a ejecutar.



PYTHON

Propiedades

- Interpretado
- Alto nivel
- Multiparadigma
- Multiplataforma
- Tipado dinámico
- Fuertemente tipado

PYTHON

Instalación

Se necesita instalar por lo menos la versión

3.7. Dependiendo del sistema operativo:



[OSX: https://docs.python-guide.org/starting/install3/osx/](https://docs.python-guide.org/starting/install3/osx/)



[Ubuntu: https://phoenixnap.com/kb/how-to-install-python-3-ubuntu](https://phoenixnap.com/kb/how-to-install-python-3-ubuntu)



[Windows: https://python-para-impacientes.blogspot.com/2017/02/instalar-python-paso-paso.html](https://python-para-impacientes.blogspot.com/2017/02/instalar-python-paso-paso.html)

VSCode

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Vamos a usar este editor de texto par hacer nuestros programas. Visual studio code tiene características como:

Tiene una consola integrada.

Tiene estilos de texto para el lenguaje Python.

Permite la instalación de nuevas características de una forma muy sencilla.

VSCode

Se puede hacer desde la pagina oficial:
<https://code.visualstudio.com>. Luego mostrar
cómo ejecutar todo lo necesario desde el
programa:

- **Abrirlo**
- **Crear un archivo con extensión “.py” para que sea un programa python.**
- **Escribir en el archivo “print(“hola”)”**
- **Abrir la terminal de VSCode**
- **Ejecutar el codigo escribiendo “python nombreDelArchivo.py” o con el botón verde de play (mostrar ambas).**
- **Ver que en la terminal se ejecuta el programa.**



Tipos Básicos en Python

Tipos Básicos: ¿Que es un Tipo de Dato?



En general, todos los lenguajes tienen una serie de Tipos de Datos que pueden manejar y con los que pueden operar.



Por ejemplo, vamos a querer representar números, y para eso cada lenguaje va a tener un tipo para representarlos. O vamos a querer representar palabras, y para eso cada lenguaje va a tener un tipo para representarlas.



TIPOS

Numéricos

int	El tipo de dato int permite representar números enteros. Su precisión es exacta.
float	El tipo de dato float permite representar números con decimales. Su precisión es aproximada.

Algunas Operaciones: +, -, *, /, //

TIPOS

Numéricos

ejemplos

Python 

```
>>> 0
>>> 2 + 2
>>> 2 * 5
>>> 3 + 2 * 3
>>> 1/4
>>> 3.123456
>>> 0.1 + 0.2
```

Python 

```
>>> 0
0
>>> 2 + 2
4
>>> 2 * 5
10
>>> 3 + 2 * 3
9
>>> 1/4
0.25
>>> 3.123456
3.123456
>>> 0.1 + 0.2
0.30000000000000004
```


TIPOS

Numéricos

ejemplos

Python 

```
>>> 10//3
>>> 10/3
>>> 10//3.4
>>> 10//3.3
>>> 10/3.4
```

Python 

```
>>> 10//3
3
>>> 10/3
3.3333333333333335
>>> 10//3.4
2.0
>>> 10//3.3
3.0
>>> 10/3.4
2.9411764705882355
```

TIPOS

Cadenas

string	El tipo de dato string , también llamado “cadena de caracteres”, permite representar una serie de caracteres concatenados (por ejemplo, palabras)
---------------	--

Algunas Operaciones: **+**, *****

TIPOS

Cadenas ejemplos

Python 

```
>>> 'Hola mundo!'
>>> "Curso introductorio :)"
>>> 'abc' + 'def'
>>> 'hola' * 3
```

Python 

```
>>> 'Hola mundo!'
'Hola mundo!'
>>> "Curso introductorio :)"
'Curso introductorio :)'
>>> 'abc' + 'def'
'abcdef'
>>> 'hola' * 3
'holaholahola'
```

TIPOS

Booleanos

bool	El tipo de dato bool , permite representar booleanos, es decir, cosas que se pueden modelar solo como Verdaderas (<i>True</i>) o Falsas (<i>False</i>).
-------------	--

Operadores de Comparación: **==** , **!=** , **<** , **<=** , **>** , **>=**

Operadores Lógicos: **and** , **or** , **not**

TIPOS

Booleanos

Operadores de Comparación

Python 

```
>>> 1 + 1 == 2
>>> 1 + 1 != 2
>>> 1 + 1 < 2
>>> 1 + 1 <= 2
>>> 1 + 1 > 2
>>> 1 + 1 >= 2
```

Python 

```
>>> 1 + 1 == 2
True
>>> 1 + 1 != 2
False
>>> 1 + 1 < 2
False
>>> 1 + 1 <= 2
True
>>> 1 + 1 > 2
False
>>> 1 + 1 >= 2
True
```

TIPOS

Booleanos

Operadores de Lógicos

Python 

```
>>> True
>>> True and False
>>> True or False
>>> not True
>>> not False
```

Python 

```
>>> True
True
>>> True and False
False
>>> True or False
True
>>> not True
False
>>> not False
True
```

TIPOS

Booleanos

Operadores de Lógicos

Python 

```
>>> 0.1  
  
>>> 0.2  
  
>>> 0.3  
  
>>> 0.1 + 0.2 == 0.3
```

Python 

```
>>> 0.1  
0.1  
>>> 0.2  
0.2  
>>> 0.3  
0.3  
>>> 0.1 + 0.2 == 0.3  
False
```

TIPOS

Booleanos

Operadores de Lógicos

Python 

```
>>> 0.1
>>> 0.2
>>> 0.3
>>> 0.1 + 0.2 == 0.3
```

Python 

```
>>> 0.1
0.1
>>> 0.2
0.2
>>> 0.3
0.3
>>> 0.1 + 0.2 == 0.3
False
```

Evitar usar == con floats debido a que son imprecisas!

Variables


¿Qué es una variable?

- ▶ Son “contenedores” que permiten guardar información
- ▶ Permiten tener memoria de los valores
- ▶ Permiten ponerle etiquetas descriptivas a los valores
- ▶ Cada dato está asociado a un tipo (ej. int, float, string, bool)



VARIABLES

Ejemplos

Python 

```
>>> x = 7
>>> y = 8
>>> x = 6
>>> x

>>> x = x + 1
>>> x

>>> y = x * x
>>> y

>>> x = 3
>>> y
```

Python 

```
>>> x = 7
>>> y = 8
>>> x = 6
>>> x
6
>>> x = x + 1
>>> x
7
>>> y = x * x
>>> y
49
>>> x = 3
>>> y
49
```

VARIABLES

Estado de computación



```
>>> x = 8
>>> y = 22
>>> lenguaje = 'Python'
```

Nombres	Valores
x	8
y	22
lenguaje	'Python'

Condiciones Booleanas

Instrucciones

Condicionales: **if**



La instrucción condicional **if** permite que se ejecuten determinadas instrucciones, sólo si una expresión se evalúa a True.



Es decir, nos permite “preguntar” si se cumple una condición o no.

```
if <expresión>:  
    <instrucciones>
```

INSTRUCCIONES CONDICIONALES

If Ejemplos

*“Si tengo sueño,
me voy a dormir”*

```
if (tengo sueño):  
    me voy a dormir
```

*“Si estoy cansado o llueve,
voy a trabajar en auto”*

```
if (estoy cansado or llueve):  
    voy a trabajar en auto
```

INSTRUCCIONES CONDICIONALES

If Ejemplos


*“**Si** la película se estrenó **y** mis amigos quieren, vamos a ir a verla”*

```
if (la película se estrenó  
and mis amigos quieren):  
    vamos a ir a verla
```

*“**Si** **no** viene el colectivo, entonces me voy a tomar un taxi”*

```
if (not viene el colectivo):  
    me voy a tomar un taxi
```


Ejercicio

5 mins 



Imprimir FizzBuzz

Implementar la lógica para que dada una variable n que representa un entero, imprima FizzBuzz si es múltiplo de 3 y de 5

Ejercicio

5 mins



Imprimir FizzBuzz

Implementar la lógica para que dada una variable n que representa un entero, imprima FizzBuzz si es múltiplo de 3 y de 5

Operador %: El operador % permite obtener el resto de una división, es importante tener en cuenta que decir ***“a es múltiplo de n”*** es equivalente a decir que ***“el resto de a dividido n es 0”*** ($a \% n == 0$).

print: La función ***print()*** permite imprimir en pantalla lo que recibe por parámetro.

```
>>> print("Hola, como estan?")
Hola, como estan?
>>> print(1)
1
>>> print(1.23456)
1.23456
```

Ejercicio

Imprimir FizzBuzz

Python 

```
if n % 3 == 0 and n % 5 == 0:  
    print('FizzBuzz')
```

Instrucciones

Condicionales: **if ... else**



La instrucción condicional **else** permite que se ejecuten determinadas instrucciones, cuando la expresión no evalúa a **True**.

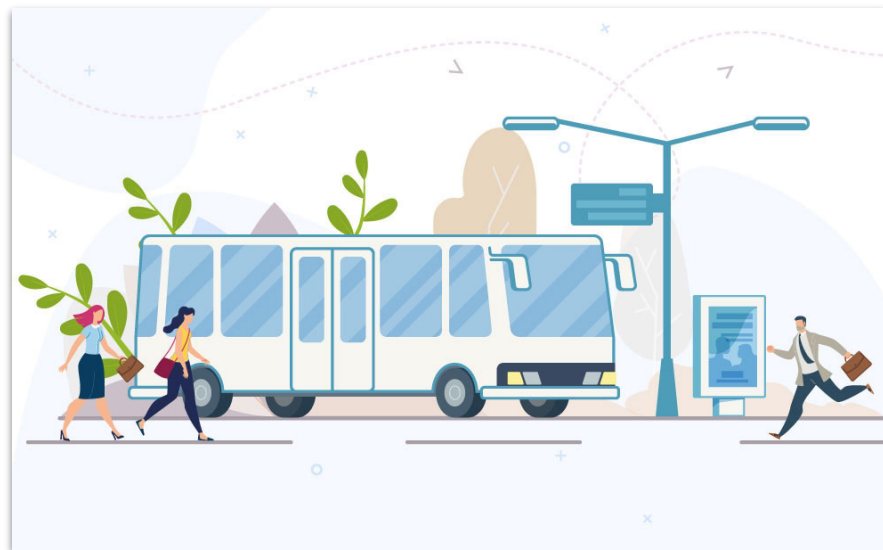
```
if <expresión>:  
    <instrucciones>  
else:  
    <instrucciones>
```

INSTRUCCIONES CONDICIONALES


If ... else Ejemplos

“Si viene el colectivo, entonces me tomo el colectivo; si no me tomo un taxi”

```
if (viene el colectivo):  
    me tomo el colectivo  
else:  
    me tomo un taxi
```



Ejercicio

5 mins 




Imprimir FizzBuzz 2.0

Implementar la lógica para que dada una variable `n` que representa un entero, se imprima FizzBuzz si es múltiplo de 3 y de 5, o el número `n` en caso contrario.

Usando el operador **if...else**

Ejercicio

Imprimir FizzBuzz 2.0

Python 

```
if n % 3 == 0 and n % 5 == 0:  
    print('FizzBuzz')  
else:  
    print(n)
```

Ejercicio

10 mins 



Imprimir FizzBuzz 3.0

Implementar la lógica para que dada una variable n que representa un entero, se imprima Fizz si es múltiplo de 3, Buzz si es múltiplo de 5 (y no es múltiplo de 3), FizzBuzz si es múltiplo de 3 y de 5, o el número n en caso contrario

Usando el operador **if...else**

Ejercicio

Imprimir FizzBuzz 3.0

Python 

```
if n % 3 == 0 and n % 5 == 0:
    print('FizzBuzz')
else:
    if n % 3 == 0:
        print('Fizz')
    else:
        if n % 5 == 0:
            print('Buzz')
        else:
            print(n)
```

Instrucciones

Condicionales: **if...elif...else**



La instrucción condicional **elif** permite que se ejecuten determinadas instrucciones, cuando no se cumplen las condiciones anteriores, y además una expresión evalúa a True.



Es equivalente a “anidar” un **if** dentro de un **else**.

```
if <expresión>:  
    <instrucciones>  
else:  
    if <expresión>:  
        <instrucciones>  
    else:  
        <instrucciones>
```

```
if <expresión>:  
    <instrucciones>  
elif <expresión>:  
    <instrucciones>  
else:  
    <instrucciones>
```

Ejercicio

10 mins



Imprimir FizzBuzz 4.0:

Implementar la lógica para que dada una variable n que representa un entero, se imprima Fizz si es múltiplo de 3, Buzz si es múltiplo de 5 (y no es múltiplo de 3), FizzBuzz si es múltiplo de 3 y de 5, o el número n en caso contrario

Usando el operador **if...elif..else**

Ejercicio

Imprimir FizzBuzz 3.0

Python 

```
if n % 3 == 0 and n % 5 == 0:  
    print('FizzBuzz')  
elif n % 3 == 0:  
    print('Fizz')  
elif n % 5 == 0:  
    print('Buzz')  
else:  
    print(n)
```

Ciclos



Ciclos

Definición: Estructura que nos permite ejecutar una serie de instrucciones muchas veces.

Ciclos

definidos: **for ... in ...**



Los ciclos definidos sirven cuando sabemos de antemano cuántas veces queremos iterar, es decir, cuántas veces queremos ejecutar las instrucciones.

```
for <nombre> in <expresión>:  
    <instrucciones>
```

CICLOS DEFINIDOS

Ejemplos

Multiplicar por 5:

Si quisiéramos multiplicar un número N por 5 y almacenar su resultado en una variable M, podríamos implementar la siguiente lógica.

$$M = 0$$
$$M = M + N$$
$$M = M + N$$
$$M = M + N$$
$$M = M + N$$
$$M = M + N$$

CICLOS DEFINIDOS

Ejemplos

Multiplicar por 5:

Si quisiéramos multiplicar un número N por 5 y almacenar su resultado en una variable M , podríamos implementar la siguiente lógica.

```
M = 0
M = M + N
M = M + N
M = M + N
M = M + N
M = M + N
```

Sabemos que queremos **sumar 5 veces N** , tenemos entonces que **iterar 5 veces**, y ejecutar cada vez la **instrucción $M = M + N$** .

CICLOS DEFINIDOS

Ejemplos

Multiplicar por 5:

Si quisiéramos multiplicar un número N por 5 y almacenar su resultado en una variable M, podríamos implementar la siguiente lógica.


```
M = 0
M = M + N
M = M + N
M = M + N
M = M + N
M = M + N
```

```
M = 0
for i in range(5):
    M = M + N
return M
```

La expresión **range(n)** nos va devolviendo todos los números del 0 al n (sin incluirlo). En este caso, se le va asignando a la variable i los valores 0, 1, 2, 3, 4.

Sabemos que queremos **sumar 5 veces N**, tenemos entonces que **iterar 5 veces**, y ejecutar cada vez la **instrucción M = M + N**.

Ejercicio

10 mins 




Sumatoria:

Implementar la lógica que permita almacenar en una variable llamada **suma** la suma de todos los números desde 0 hasta una variable **n** dada.

Ejercicio

Sumatoria

Python 

```
suma = 0
for i in range(n+1):
    suma += i
```

CICLOS INDEFINIDOS

While

Los ciclos indefinidos sirven cuando no sabemos de antemano cuántas veces queremos iterar, es decir, cuántas veces queremos ejecutar las instrucciones.

Nota: En general, los ciclos que se pueden hacer con un **for**, también se pueden hacer con un **while**, pero no necesariamente el **while** se puede reemplazar con un **for**.

```
while <expresión>:  
    <instrucciones>
```

EJEMPLO

Multiplicar por 5

```
M = 0
M = M + N
M = M + N
M = M + N
M = M + N
M = M + N
```

Sin ciclos


```
M = 0
for i in range(5):
    M = M + N
```

Ciclo definido

```
M = 0
i = 0
while i < 5:
    M = M + N
    i += 1
```

Ciclo indefinido

Ejercicio

10 mins 



Imprimir Números:

Implementar la lógica para imprimir todos los números entre 0 y una variable N dada.

- a) Haciendo uso de un ciclo **definido**.
- b) Haciendo uso de un ciclo **indefinido**.

Ejercicio

Imprimir Números

Python 

```
for i in range(N):  
    print(i)
```

```
i = 0  
while i < N:  
    print(i)  
    i += 1
```


Ejercicio



Suma de fizz y buzz:

Si tomamos todos los números menores a 10 que son múltiplos de 3 o 5, obtenemos 3, 5, 6 y 9. La suma entre ellos da 23.

Utilizando lo aprendido en FizzBuzz y usando ciclos definidos, encontrar la suma de todos los múltiplos de 3 o 5 menores a 1000.