# Conociendo Python (parte II)

Tutoría

{desafío} latam_

# Ideas fuerza

## *Conociendo Python*

***Un algoritmo** es una **secuencia ordenada y finita de pasos** que entregan una **respuesta o solución.** Se puede representar mediante un **diagrama de flujo.***

*Utilizamos **operadores** de diferente tipo para asignar, realizar cálculos, verificar y/o comparar variables.*

*Los **ciclos** nos permiten realizar procesos repetitivos. En Python utilizamos **for** y **while**.*

# Recursos asincrónicos

- ¿Revisaste los recursos asincrónicos?
- ¿Alguno de ellos te dejó dudas?

# /*Algoritmos*/

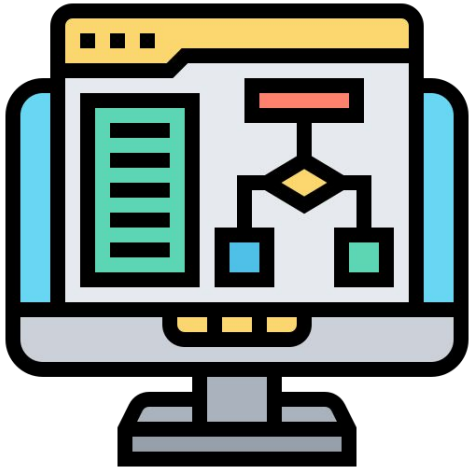
# Algoritmo

## *¿Qué es un algoritmo?*

Se llama **algoritmo** a una serie de pasos **ordenados** y **bien definidos** que se utilizan para resolver un problema específico.

Es como una receta que sigue un cocinero para preparar un plato de comida, pero en lugar de cocina, se utiliza en programación para resolver problemas.

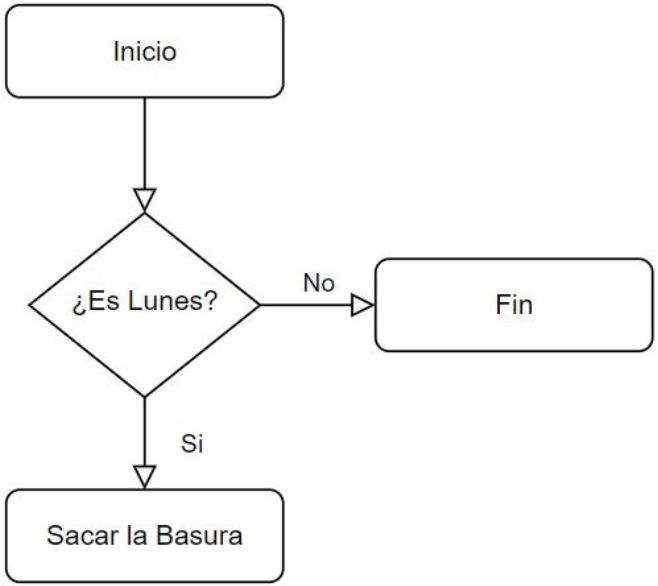
- Debe entregar, efectivamente, una solución o respuesta.
- Debe ser finito, es decir, terminar (esto no es trivial).
- Debe ser preciso, es decir, no prestarse a interpretaciones.



# /*Diagramas de flujo*/

# Diagrama de flujo

*Tomando decisiones*




- **Rectángulos:** se utilizan para representar procesos o actividades.
- **Rombos:** se utilizan para representar decisiones, es decir, puntos en el proceso en los que se debe tomar una decisión basada en cierta condición.
- **Flechas:** se utilizan para conectar las diferentes figuras geométricas y mostrar la dirección del flujo del proceso.

# /*Operadores*/

# Operadores de asignación y comparación

```
a=5
b=10

print(a==b)
print(a!=b)
print(a>b)
print(b>=a)
print(b>=7>a)
```

```
False
True
False
True
True
```

# Operadores lógicos

***and***

```
print("True and True:",True and True)
print("True and False:",True and False)
print("False and False:",False and False)
```

```
True and True: True
True and False: False
False and False: False
```

# Operadores lógicos

*or*

```
print("True or True:",True or True)
print("True or False:",True or False)
print("False or False:",False or False)
```

```
True or True: True
True or False: True
False or False: False
```

# Operadores lógicos

*not*

```
print("not True:", not True)
print("not False:", not False)
```

```
not True: False
not False: True
```

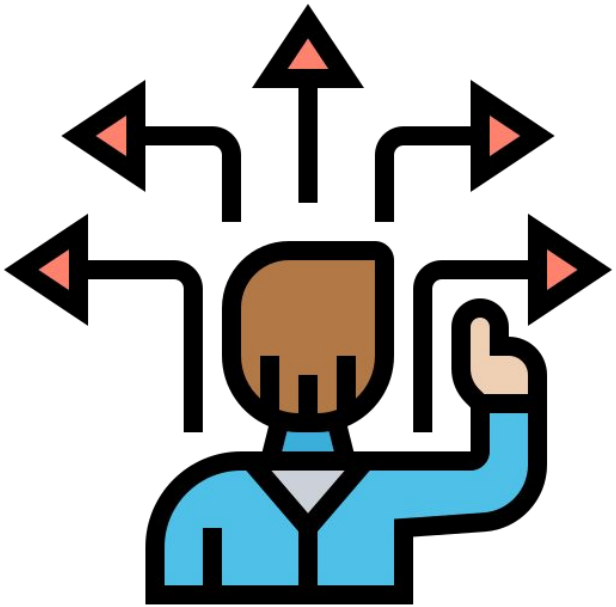
# /*Estructuras de control*/

# Estructuras de control

## *if - else - elif*

Los **comandos** "if - else - elif" en Python permiten tomar decisiones basadas en una o varias condiciones, lo que permite que el programa pueda actuar de manera diferente según el caso que se presente.

-



{desafío} latam_

# Estructuras de control

*Indentación*

```
p = 7
if p > 5:
    print("azulejo")
    print(p+1)
```

```
azulejo
8
```

# Estructuras de control

## *¡Practiquemos!*

Veremos directamente, en Jupyter Notebook, el uso de las estructuras de control.

1. if
2. else
3. elif

{desafío} latam_

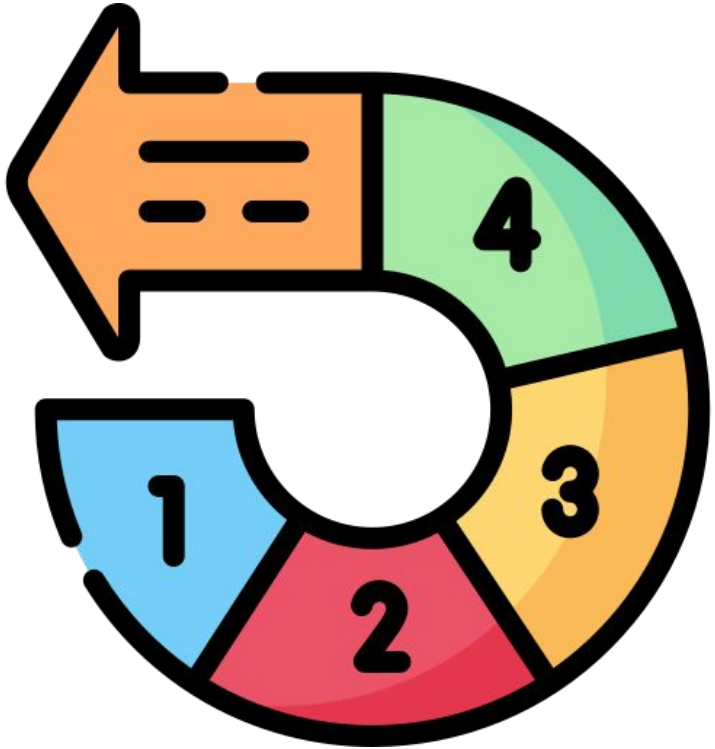
# /*Ciclos*/

# Ciclos

## *Repetición de procesos*

Los **ciclos** en programación son formas de repetir una tarea varias veces, sin hacerlo manualmente.

- for
- while



# Ciclos

*Estructura*

```
lista = ["a",3,"t","perro"]
for i in lista:
    print(i)
```

```
a
3
t
perro
```

```
p = 0

while p <= 3:
    print(p)
    p = p + 1
```

```
0
1
2
3
```

# Ciclos

## *¡Practiquemos!*

Veremos directamente, en Jupyter Notebook, el uso de los ciclos. Abordaremos:

1. ciclo for
2. ciclo while
3. ciclos anidados

{desafío} latam_

# Desafío - Conociendo Python II


{desafío} latam_

# Desafío

## *“Conociendo Python II”*

- ¿Hay contenidos que necesitas repasar antes de comenzar el desafío?
- ¿Comprendes bien qué te están solicitando en cada caso?

{desafío}
latam_

# Próxima sesión...

- *Utilizar y operar datos con diferentes estructuras.*
- *Cargar diferentes tipos de datos.*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales