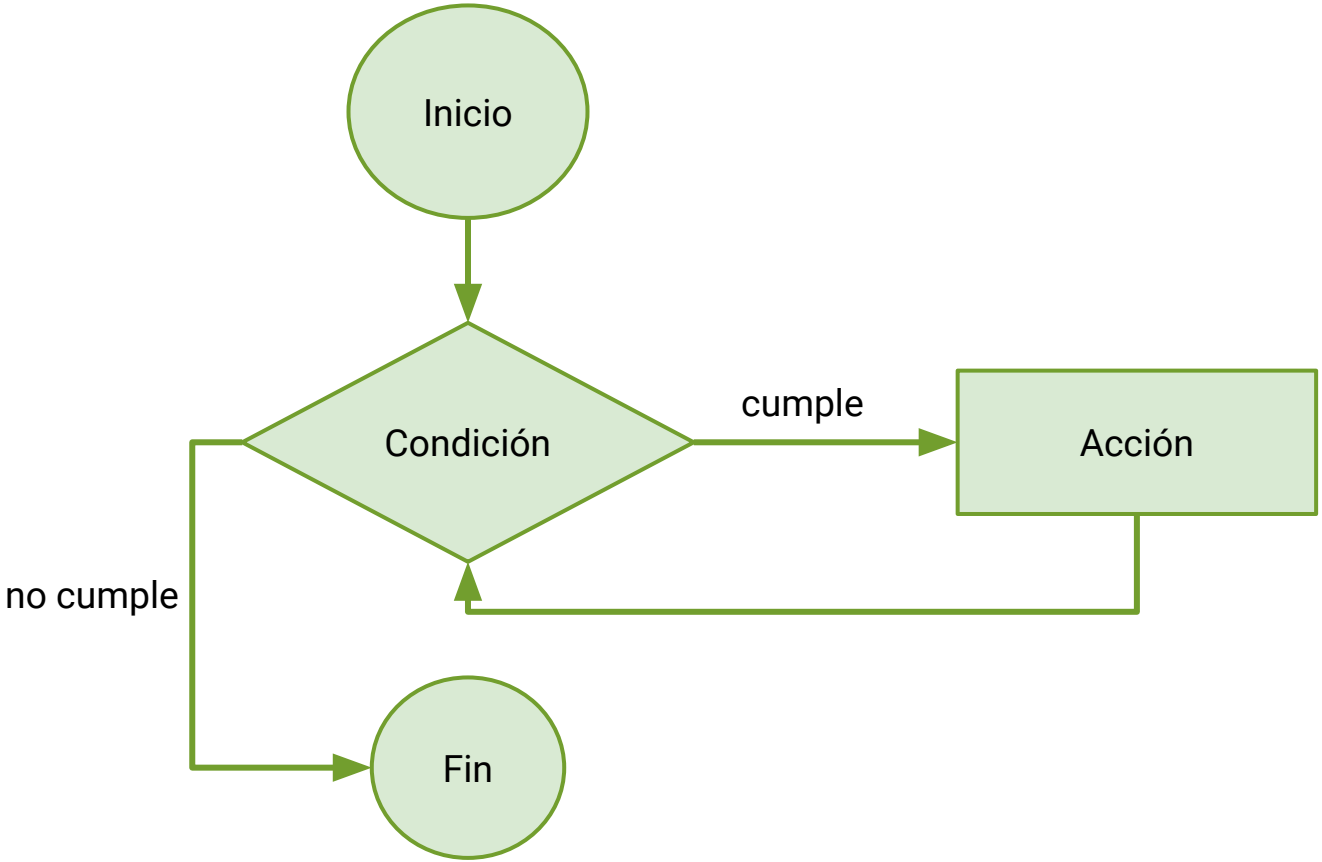
Ciclos y métodos _

RECETA DE PANQUEQUES	ALGORITMO
- Agregar 1 taza de harina en un bowl - Agregar 1 taza de leche a la harina - Agregar 1 huevo a los ingredientes previos - Revolver y mezclar los 3 ingredientes - Precalentar el sartén - Agregar parte de la mezcla hasta cubrir el sartén y esparcir una capa delgada - Esperar 1 minuto - Dar vuelta la masa - Esperar otro minuto - Retirar el panqueque - **Si queda masa**, agregar parte de la mezcla hasta cubrir el sartén y esparcir una capa delgada - Esperar 1 minuto ...	1. Agregar 1 taza de harina en un bowl. 2. Agregar 1 taza de leche a la harina. 3. Agregar 1 huevo a los ingredientes previos. 4. Revolver y mezclar los 3 ingredientes. 5. Precalentar el sartén. 6. Agregar parte de la mezcla hasta cubrir el sartén y esparcir una capa delgada. 7. Esperar 1 minuto. 8. Dar vuelta la masa. 9. Esperar otro minuto. 10. Retirar el panqueque. 11. **Repetir pasos del 6 al 10 hasta terminar la mezcla.**


Inicio
Condición
cumple
Acción
no cumple
Fin

Validar entrada de un usuario utilizando **while**

```python
# Primero se pide la entrada al usuario
# IMPORTANTE: La variable que se utilice para evaluar la condición DEBE
# estar definida antes de que se ejecute el ciclo
number = int(input("Ingresa un número entre 1 y 10"))

# Después se inicia el ciclo
# La condición de salida es que el número esté entre 1 y 10
while number < 1 or number > 10:
    # Este código se ejecutará "mientras" se ingrese un número
    # menor a 1 o mayor a 10
    print("El número ingresado no está entre 1 y 10")
    number = int(input("Ingresa un número entre 1 y 10"))

print("El número ingresado fue ", number)
```

Ejercicio:
Menú de opciones

```python
# Se inicializa la variable para evaluar la condición de while
opcion = ""

# La condición de salida será que opción sea "salir"
while opcion != "salir":

    # Se agregan los print para dibujar las opciones del menú
    print("Ingrese una opción")
    print("1: Sumar 2 + 2")
    print("2: Multiplicar 2 * 2")
    print("salir: Salida")

    # Se solicita la opción al usuario
    opcion = input()

    # Si escoge 1, se hará la suma 2 + 2
    # opcion vale 1, por lo que volverá a entrar en el ciclo
    if opcion == "1":
        print(2 + 2)

    # Si escoge 2, se hará la multiplicación 2 * 2
    # opcion vale 2, por lo que volverá a entrar en el ciclo
    elif opcion == "2":
        print(2 * 2)

    # Si escoge "salir", se muestra el mensaje indicando que se detendrá el ciclo
    # opción ha tomado el valor de la condición de salida (no se volverá a mostrar el menú)
    elif opcion == "salir":
        print("Saliendo")
```

Operadores de asignación

OPERADOR	NOMBRE	EJEMPLO	RESULTADO
=	Asignación	a = 2	**a** toma el valor **2**
+=	Incremento y asignación	a += 2	**a** es incremento en **2** y asignado el valor resultante
-=	Decremento y asignación	a -= 2	**a** es reducido en **2** y asignado el valor resultante
*=	Multiplicación y asignación	a *= 3	**a** es multiplicado por **3** y asignado el valor resultante
/=	División y asignación	a /= 3	**a** es dividido por **3** y asignado el valor resultante

Sumatoria
1 a N

```python
# Importar módulo sys
import sys

# Almacenar en una variable el valor ingresado como argumento
# Debe guardarse como int
limite = int(sys.argv[1])

# Crear e inicializar en 0 la variable "i" que funcionará como contador
i = 0

# Crear e inicializar en 0 la variable "suma" que irá acumulando la sumatoria
suma = 0

# Crear el ciclo while que iterará mientras el contador sea menor al límite
while i < limite:
    # Aumentar en 1 el valor del contador en cada iteración
    i += 1

    # Acumular en la variable suma el valor de i en cada iteración
    suma += i

# Imprimir el valor final de la sumatoria al salir del ciclo
print(suma)
```

Lista HTML

Paso 1: Código inicial

```python
# Importar el módulo sys
import sys

# Almacenar en una variable el valor ingresado como argumento
# Debe guardarse como int
items = int(sys.argv[1])

# iniciar el conador en 0
i = 0

# iniciar el acumulador como string vacío
html = ""

# Iterar mientras el contador sea menor a items
while i < items:
    # aumentar el contador en 1 en cada iteración
    i += 1
    # Concatenar los tag de apertura y cierre para cada elemento de la lista
    html += "<li></li>"

print(html)
```

Paso 2: Agregar contenido a cada elemento de la lista

```
html += “<li>Elemento</li>”
```

Paso 3: Interpolar el valor del iterador en el contenido

```
html += “<li>Elemento {}</li>”.format(i)
```

Paso 4: Agregar salto de línea

```
html += “<li>Elemento {}\n</li>”.format(i)
```

Paso 5: Agregar tags de apertura y cierre de la lista

```
11  # Agregar tag de apertura de la lista
12  html = "<ul>\n"
13
14  # Iterar mientras el contador sea menor a items
15  while i < items:
16      # aumentar el contador en 1 en cada iteración
17      i += 1
18      # Agregar salto de línea
19      html += "<li>Elemento {}\n</li>".format(i)
20
21  # Agregar tag de cierre de la lista
22  html += "</ul>"
23  print(html)
```

Paso 6: Agregar tabulación en los elementos de la lista

```
html += “\t<li>Elemento {}\n</li>”.format(i)
```

Patrón
“nick msn”

Posición	0	1	2	3	4	5	6	7	8	9	10	11	12
Símbolo	★	·.·	‾	·.·	★	·.·	‾	·.·	★	·.·	‾	·.·	★

0	1	2	3	4	5	6	7	8	9	10	11	12
★	·.·	‾	·.·	★	·.·	‾	·.·	★	·.·	‾	·.·	★

0	1	2	3	4	5	6	7	8	9	10	11	12
★	·.·	‾	·.·	★	·.·	‾	·.·	★	·.·	‾	·.·	★

Casos

- Impar: 
- Divisible por 4: 
- El resto: 

Código final

```python
#Importamos sys
import sys

# Almacenamos el argumento como int
limit = int(sys.argv[1])

#Patrón de referencia
#pattern = "★·.·´¯`·.·★·.·´¯`·.·★·.·´¯`"

# almacenamos cada elemento del patrón en una variable
a = "★"
b = "·.·"
c = "´¯`"

# inicializamos el acumulador
contain = ""

# Recorremos el rango hasta el límite más 1
for i in range(limit + 1):
    # Primer caso: Concatenar puntos si el elemento es inpar
    if i % 2 != 0:
        contain += b

    # Segundo caso: Concatenar estrella si el elemento es divisible por 4
    elif i % 4 == 0:
        contain += a

    # El resto serán las líneas
    else:
        contain += c

# Se imprime el acumulador
print(contain)
```

Desafío
Lluvia de agua nieve

```python
# Importamos sys
import sys

# Importamos random
import random

# Almacenamos el ancho como int
width = int(sys.argv[1])

# Validamos que no sea menor a 10
if width < 10:
    width = 10

# Creamos el contenedor
output = ""

# Recorremos de 1 a 9 en el loop externo
for i in range(1, 10):
    # Creamos el primer número al azar, utilizando el iterador y el ancho del usuario
    rand_number = random.randint(i, width)

    # Usamos el número creado para agregar espacios en blanco antes de la nieve
    output += " " * rand_number + "*" + "\n"

    # Recorremos el segundo loop con rango de 1 a i
    for j in range(1, i):
        # Creamos el segundo número al azar, utilizando el segundo iterador y el ancho
        rand_number_2 = random.randint(j, width)

        # Concatenamos la lluvia siguiento la misma lógica del loop externo
        output += " " * rand_number_2 + "/" + "\n"

# Mostramos el lienzo creado
print(output)
```

Desafío
Nube y lluvia

(basado en el desafío agua y nieve)

Paso 1: Eliminar identación, segundo for y cambiar range

```
for i in range(1, 10):
    # Creamos el primer número al azar, utilizando el iterador y el ancho del usuario
    rand_number = random.randint(i, width)

    # Usamos el número creado para agregar espacios en blanco antes de la nieve
    output += " " * rand_number + "*" + "\n"

# Recorremos el segundo loop con rango de 1 a 10
for j in range(1, 10):
    # Creamos el segundo número al azar, utilizando el segundo iterador y el ancho
    rand_number_2 = random.randint(j, width)

    # Concatenamos la lluvia siguiento la misma lógica del loop externo
    output += " " * rand_number_2 + "/" + "\n"

```

Paso 2: Transformar la nieve en nube

```
18  for i in range(1, 10):
19      # Creamos el primer número al azar, utilizando el iterador y el ancho del usuario
20      rand_number = random.randint(i, width)
21
22      # Cambiar nieve por nube
23      output += "@" * rand_number + "\n"
24
```

Paso 3: Iniciar contenedor con @

```
14  # Inicializamos contenedor con el ancho de la nube
15  output = "@" * width + "\n"
16
```

Paso 4: Cambiar lógica para crear primer random

```
18  for i in range(1, 10):
19      # Creamos inicio del random como el 80% de width
20      start_random = int(0.8 * width)
21
22      # Creamos el primer número al azar, utilizando el start_random y el ancho del usuario
23      rand_number = random.randint(start_random, width)
24
25      # Cambiar nieve por nube
26      output += "@" * rand_number + "\n"
27
```

Llamar vs Definir

- Cuando se **crea** una función, ésta se está definiendo.

- Cuando se **usa** una función, ésta se está llamando.

Parámetros

- Variables locales
- Se utilizan en la lógica de la función
- Se definen junto con la función

Tipos de variables según su alcance

- Globales
- Locales
- De instancia
- De clase