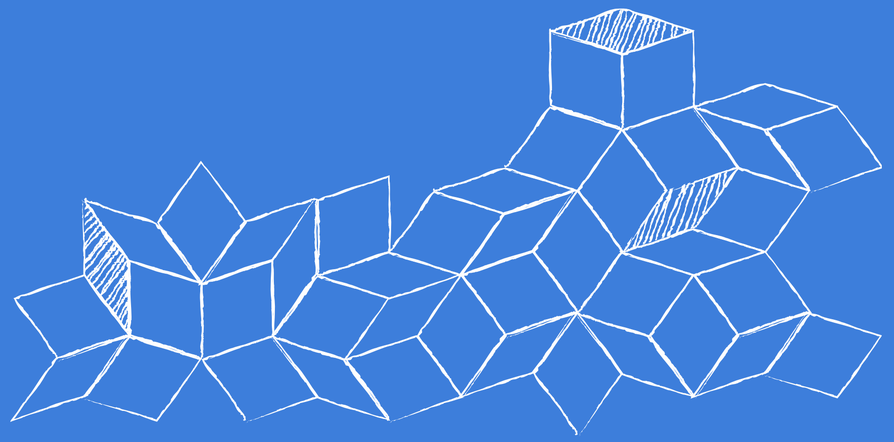
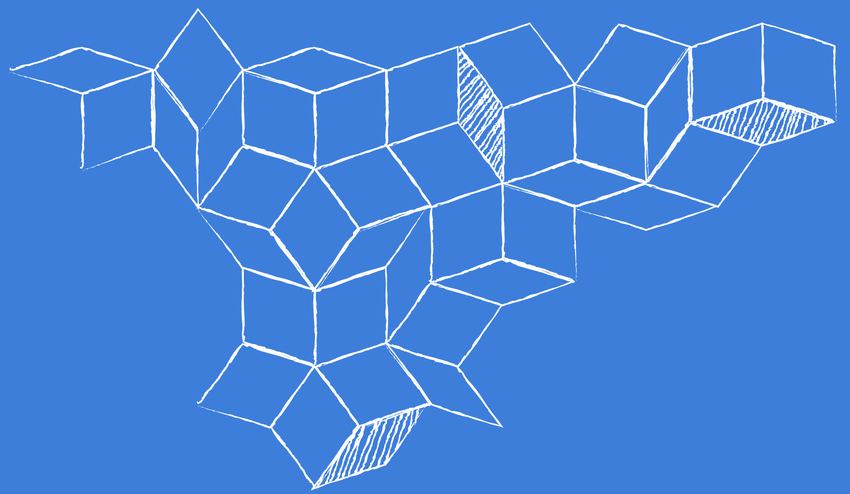


STEMUC

Academia para **escolares**



Suma MAT UC

preingeniería

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

PRIMEROS PASOS
EN PROGRAMACIÓN

Clase 4

Instrucciones repetitivas y más librerías

Ya hemos visto casi toda la materia 😱

Repasémoslo un poco lo de ayer

Funciones existentes: `import`

La instrucción `import`, nos permite usar código definido en otros archivos llamados módulos o librerías. Los módulos se cargan en memoria mediante importación.

```
import nombre_módulo

nombre_módulo.nombre_función(…)
```

```
from nombre_módulo import nombre_función

nombre_función(…)
```

Funciones existentes: `random`

La librería `random`, que nos entrega funciones que producen números aleatorios.

```
import random

print(random.random())       # float entre 0 y 1 (inclusive)
print(random.randint(1,2))   # int entre 1 y 2 (inclusive)
```

```
from random import random, randint

print(random())      # float entre 0 y 1 (inclusive)
print(randint(1,2)) # int entre 1 y 2 (inclusive)
```

Control de flujo: ciclos o *loops*

En muchos programas es necesario repetir operaciones o líneas de código. Casos típicos incluyen:

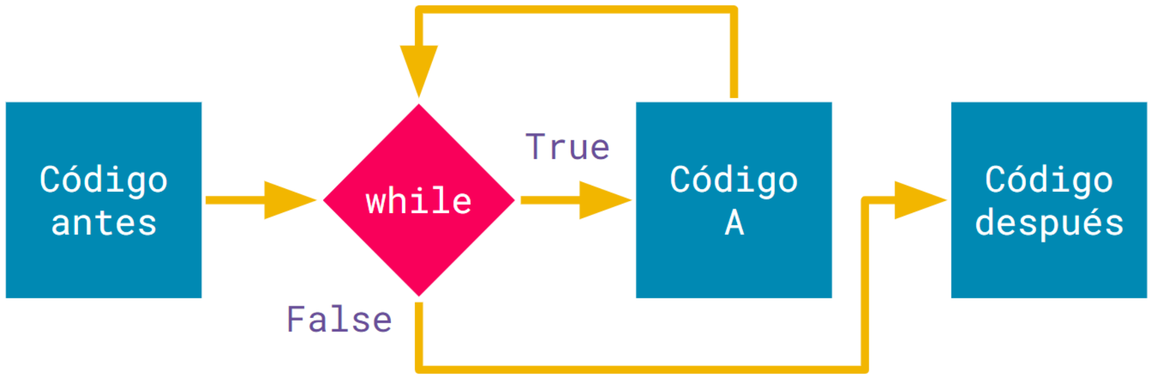
- Recibir una cantidad indefinida de *inputs*.
- Calcular algo hasta llegar a un resultado.
- Ejecutar operaciones hasta que se indique que el programa debe cerrarse.

Para ello, Python cuenta con dos comandos primitivos de ciclo:

1. `while` Permite ejecutar una serie de líneas mientras se cumpla una condición.
2. `for` Permite iterar sobre una secuencia.

Ciclos: `while`

La instrucción `while` permite ejecutar varias veces la misma sección de código.




```
# Código antes
while condición:
    # Código A
    # …

# Código después
```

Importante: Necesitamos que el código modifique la condición para poder salir del ciclo, sino el código nunca terminará.

Hagamos un ejercicio

Para calentar la materia

Problemas

- Sucesión matemática.
  - Escribe un programa que pida un entero n y entregue los primeros n elementos de la sucesión:

    $1, 3, 5, 7, 9, \ldots$

- Partido de futbol.
  - Escriba un programa que registre goles, dependiendo si es “local” o “visita”. El partido termina a los 5 goles.

Ahora sí,

¡Veamos la última materia del curso!

En la clase de hoy…

- Aprenderemos las instrucciones `for` e `in range`.
  Con ellos podremos manejar los **flujos repetitivos** un número **determinado** de veces.

- Conocerán nuevos módulos de Python:
  - `calendar` y `datetime`: Con ellos manejaremos aspectos relacionados a fechas y el tiempo.
  - `time`: Nos permite manejar el flujo del programa.
  - `math`: Nos da acceso constante y operaciones matemáticas.

Control de flujo: ciclos o *loops*

En muchos programas es necesario repetir operaciones o líneas de código. Casos típicos incluyen:

- Recibir una cantidad indefinida de *inputs*.
- Calcular algo hasta llegar a un resultado.
- Ejecutar operaciones hasta que se indique que el programa debe cerrarse.

Para ello, Python cuenta con dos comandos primitivos de ciclo:

1. `while` Permite ejecutar una serie de líneas mientras se cumpla una condición.
2. `for` Permite iterar sobre una secuencia.

Ciclos: `for`

La instrucción `for` también permite ejecutar varias veces la misma sección de código, pero combinada con la instrucción `in range` podemos repetirla un número fijo de veces.

```
for variable in range(inicial, final, step):
    # Código_for
    # ...

# Código fuera del for
```

Nota: Con `for`, no necesitamos que el código modifique la condición para poder salir del ciclo.

Ciclos: `for`

La instrucción `for` también permite ejecutar varias veces la misma sección de código, pero combinada con la instrucción `in range` podemos repetirla un número fijo de veces.

```
n = 0
while n < 5:                    # Mientras i < 5
    print('Hola')               # Imprimimos
    i = i + 1                   # Aumentamos el contador
```

```
for n in range(0, 5, 1):    # Contamos hasta 5
    print('Hola')           # Imprimimos
```

Rangos de números: `range`

La instrucción `range` nos entrega una secuencia de números. Podemos obtener estos rangos de tres formas distintas:

1. `range(f)` Indicando solo el final (`f`), entrega un rango entre `0` y `f - 1`.
2. `range(i, f)` Indicando el inicio (`i`) y el final (`f`), entrega un rango entre `i` y `f - 1`.
3. `range(i, f, s)` Indicando el inicio (`i`), final (`f`) y los saltos (`s`), entrega un rango entre `i` y `f - 1`, con saltos de porte `s`.

Rangos de números: `range`

La instrucción `range` nos entrega una secuencia de números. Podemos obtener estos rangos de tres formas distintas:

```
for i in range(5):          # Rango: 0, 1, 2, 3, 4
    print(i)

for i in range(2, 7):       # Rango: 2, 3, 4, 5, 6
    print(i)

for i in range(3, 12, 3): # Rango: 3, 6, 9
    print(i)
```

Ciclos: `for`

Usando `for`, podemos aprovechar y darle utilidad a las variables que están siendo iteradas y modificadas.

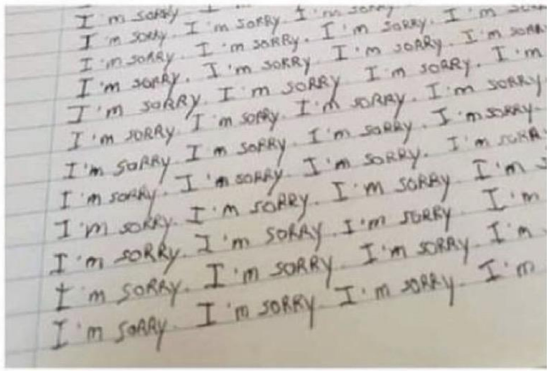
```
for i in range(3):
    for j in range(i + 1):
        print("Valor i:", i, "| Valor j:", j)
```

```
Valor i: 0 | Valor j: 0
Valor i: 1 | Valor j: 0
Valor i: 1 | Valor j: 1
Valor i: 2 | Valor j: 0
Valor i: 2 | Valor j: 1
Valor i: 2 | Valor j: 2
```

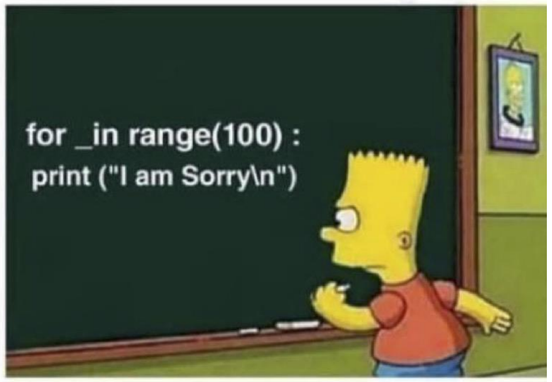
Ciclos: `for`

Teacher : Write " I am Sorry "
100 times as punishment .

Normal Students



Programmer



Ciclos: `for`

Además de recorrer secuencias de números, el `for` también nos permite recorrer secuencias de caracteres, es decir los *strings*.

```
for variable in str:
    # Código_for
    # ...

# Código fuera del for
```

```
for carácter in 'Hola mundo':
    print(carácter)   # H, o, l, a, , …
```

Veamos unos ejemplos

Funciones existentes.

Entre las múltiples librerías *built-in* que tiene Python, ayer aprendimos `random`, pero existen muchas otras más:

- `datetime` Nos permite manejar fechas.
- `calendar` Nos permite obtener calendarios.
- `time` Nos permite manejar el flujo del programa.
- `math` Nos entrega acceso a constantes y operaciones matemáticas más complejas.

Funciones existentes: `datetime`

```python
import datetime

# Obtener el día de hoy
hoy = datetime.date.today()
print(hoy)

# Obtener la fecha y hora actual
ahora = datetime.datetime.now()
print(ahora)

# Obtener la fecha de un día dado.
# Se especifica año, mes, día; en dicho orden.
fecha_específica = datetime.datetime(2022, 12, 28)
print(fecha_específica)
```

Funciones existentes: `calendar`

```
import calendar

# Podemos obtener el calendario de
# un mes y año en específico
mes = 1
año = 2024
print(calendar.month(año, mes))

# U obtener el calendario de un año completo
print(calendar.calendar(2024))
```

Funciones existentes: `time`

```
import datetime
import time

print('Inicio:', datetime.datetime.now())
time.sleep(3)    # Esperaremos 3 segundos antes de
                 # pasar a la siguiente instrucción
print('Fin:   ', datetime.datetime.now())
```

Funciones existentes: `math`

```
import math

# Podemos acceder a constantes matemáticas
print('Pi:        ', math.pi)
print('e:         ', math.e)
print('Infinito: ', math.inf)

# También tenemos más operaciones matemáticas
n = math.e
print('Techo:           ', math.ceil(n))
print('Piso:            ', math.floor(n))
print('Truncar:         ', math.trunc(n))
print('‘Raíz cuadrada: ', math.sqrt(n))
print('Logaritmo:       ', math.log(n, 2))
# NOTA: El segundo número corresponde a la base
# del logaritmo
```

Es momento de  empezaren sus proyectos finales

1. En grupos de 2-3 personas, hagan un programa.
2. El tema es libre, por lo que pueden hacerlo de lo que más les guste.

Nota: Si no se les ocurre qué hacer, pueden basarse en alguno de los problemas propuestos.

¿Qué es lo que aprendieron hoy?

Ve a

www.menti.com

Introduce el código

8532 2274