Clase 03 - Python: variables y expresiones

IIC1103-07 - Introducción a la Programación

Cristian Ruz – `cruz@ing.puc.cl`

Martes 13-Agosto-2019

Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

Contenidos

Contenidos

Contacto

- `iic1103@uc.cl`. Nicolás Quiroz
  - Coordinación
  - Justificación de inasistencias
  - Asistencia mal contada
  - Notas incorrectas en planilla
  - Recorrecciones
- `cruz@ing.puc.cl`. Profesor.
  - Materia
  - Situaciones especiales
- `rsalvarez@uc.cl`. Raúl Álvarez. Ayudante
  - Materia
  - Notas de participación
  - a.k.a. `KnowYourselves`

Contenidos

Laboratorios

Laboratorios: desde el **Lunes 27-Agosto**

- Laboratorios evaluados automáticamente (*hackerrank*)
- Pueden ir a cualquier laboratorio (Lunes a Jueves)
  - SIN LAPTOP
    - Lunes a Jueves, mód 5 y 6: Lab San Agustín (piso 2)
  - CON LAPTOP
    - Martes, mód 5 y 6: K200, A5, B12
    - Miércoles, mód 5 y 6: **B23**, CS203, C203
    - Jueves, mód 5 y 6: B13, K204, CS101
- **¡Este es un curso práctico! Aprender haciendo**

Problema 1: Escribir en pantalla

Problema 1: Cree un programa que escriba en pantalla el mensaje "*Alerta de emergencia. Mensaje de prueba sin costo*".

Algoritmo:

1. **Escribir** en pantalla el texto: "Alerta de emergencia. Mensaje de prueba sin costo"

Problema 1: Escribir en pantalla

Problema 1: Cree un programa que escriba en pantalla el mensaje "*Alerta de emergencia. Mensaje de prueba sin costo*".


```
1 print("Alerta de emergencia. Mensaje de prueba sin costo")
```

Hemos hecho un programa que **escribe** una **salida** (*output*)

Problema 2: Preguntar, recordar y escribir

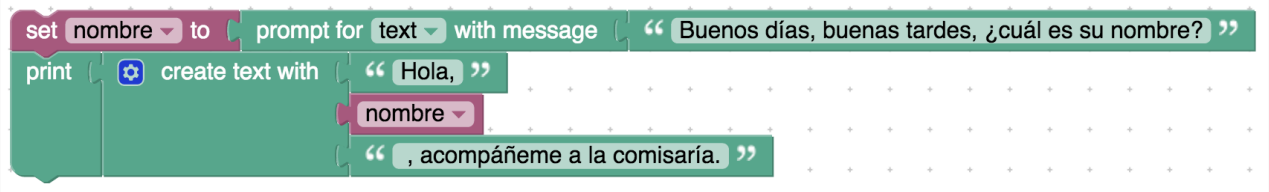
Problema 2: Cree un programa que pregunte al usuario el nombre y lo salude.

Algoritmo:

1. Preguntar el nombre de la persona
2. Recordar el nombre de la persona
3. Escribir el saludo usando el nombre de la persona

Problema 2: Preguntar, recordar y escribir

Problema 2: Cree un programa que pregunte al usuario el nombre y lo salude.




```
1 nombre = input("Buenos días, buenas tardes, ?`cuál es su nombre? ")
2 print("Hola," + nombre + ", acompáñeme a la comisaría.")
```

Hemos hecho un programa que **pregunta** un texto de **entrada** (*input*) al usuario, lo **recuerda** (en una *variable*), y luego **escribe** un texto de **salida** (*output*)

Problema 3: Operaciones matemáticas

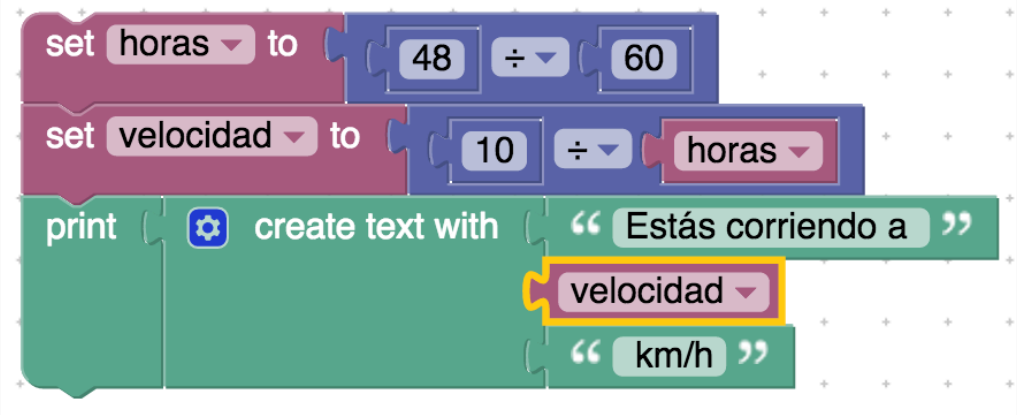
Problema 3: Cree un programa que calcule mi velocidad en km/h, si demoro 48 minutos en correr 10 kilómetros.

Algoritmo:

1. Convertir los minutos a horas.
2. Dividir los kilómetros por las horas.
3. Escribir el resultado de la división.

Problema 3: Operaciones matemáticas

Problema 3: Cree un programa que calcule mi velocidad en *km/h*, si demoro 48 minutos en correr 10 kilómetros.




```
1 horas = 48/60
2 velocidad = 10 / horas
3 print("Estás corriendo a " + str(velocidad) + " km/h")
```

Hemos hecho un programa que **calcula** un valor usando **operaciones matemáticas**, y **escribe** el resultado en un texto de **salida** (*output*)

Problema 4: Condicionales

Problema 4: Cree un programa que me ayude a saber si debo subirme al metro o no para llegar a San Joaquín.

Algoritmo:

1. Preguntar el color de la ruta actual
2. Si la ruta actual es roja, entonces debo subirme
3. Si la ruta es verde, entonces no debo subirme
4. Si no hay ningún color, entonces puedo subirme

Problema 4: Condicionales

Problema 4: Cree un programa que me ayude a saber si debo subirme al metro o no para llegar a San Joaquín.


```
1 color = input("¿De qué color es la ruta?")
2 if color.lower() == "rojo":
3     print("¡Súbete!, pero deja bajar antes de subir.")
4 elif color.lower() == "verde":
5     print("No te subas. No para en San Joaquín.")
6 else:
7     print("¡Súbete!, pero deja bajar antes de subir.")
```

Problema 4: Condicionales

```
color = input("¿De qué color es la ruta?")
if color.lower() == "rojo":
    print("¡Súbete!, pero deja bajar antes de subir.")
elif color.lower() == "verde":
    print("No te subas. No para en San Joaquín.")
else:
    print("¡Súbete!, pero deja bajar antes de subir.")
```

Hemos hecho un programa que ejecuta *una acción u otra* (i.e. toma una **decisión**), dependiendo de una **condición lógica** (*booleana*).

Problema 5: Repeticiones (iteraciones)

Problema 5: Voy en el metro hacia Baquedano. Cree un programa que me avise si debo bajarme del metro.

Algoritmo:

1. Preguntar la estación actual
2. Mientras no sea Baquedano:
    2.1 No debo bajarme
    2.2 Volver a preguntar la estación
3. Si es Baquedano, indicar que debo bajar.

Problema 5: Repeticiones (iteraciones)

Problema 5: Voy en el metro hacia Baquedano. Cree un programa que me avise si debo bajarme del metro.


```
1 estacion = input("¿En qué estación estás?")
2 while estacion.lower() != "baquedano":
3     print("Aún no has llegado. ¡No te bajes!")
4     estacion = input("¿En qué estación estás?")
5 print("Has llegado a Baquedano. ¡Bájate!")
```

Hemos hecho un programa que **repite** una (o más) acción(es), dependiendo de una **condición lógica** (*booleana*).

¡Y eso es!

TODOS los programas que vamos a escribir (y que existen) se pueden construir con:

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

Esto entra para la I1:

- Lectura (`input`) y escritura (`print`) de datos
- Uso de **variables** para recordar datos
- Cómputo de **expresiones** usando operaciones matemáticas
- Instrucciones **condicionales**: `if`, `elif`, `else`
- Instrucciones **cíclicas**: `while`, `for`
- ...y funciones ...

Contenidos

Al final del curso sabremos ...

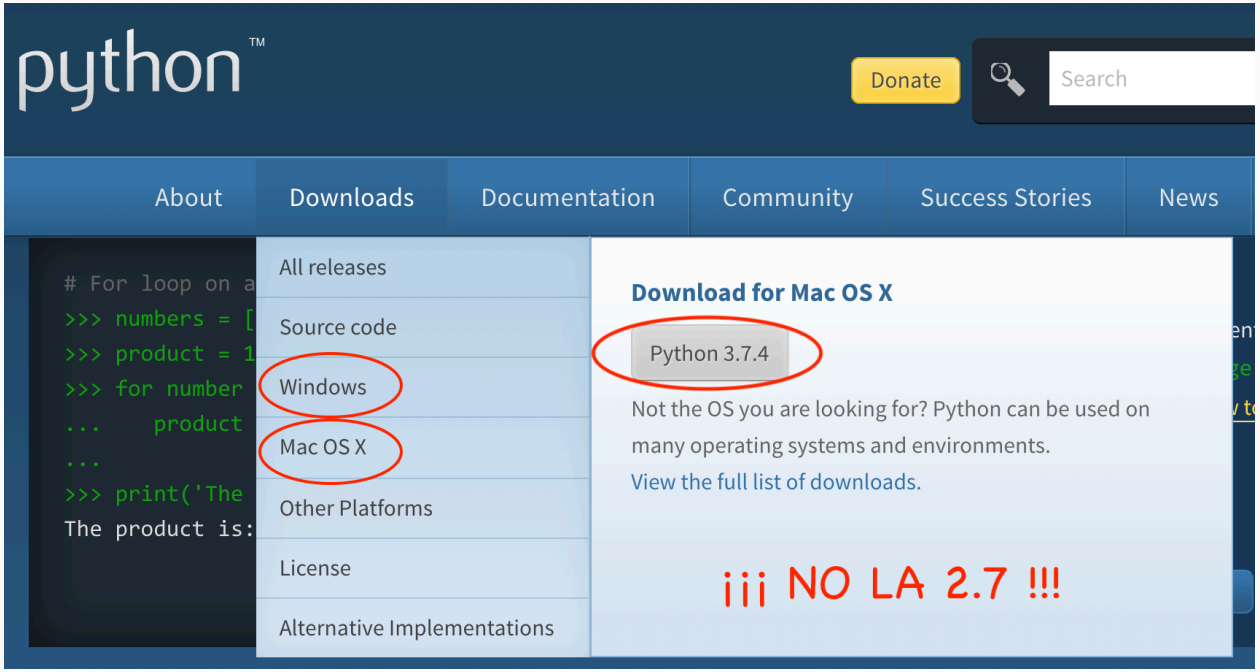
1. **Variables y expresiones**
2. Control de flujo
3. Funciones y recursión
4. Strings
5. Listas
6. Tipos de datos personalizados (objetos)
7. Ordenación y búsqueda
8. Archivos

Contenidos

El Lenguaje

Python 3.7.x.

`https://www.python.org/`



¿Lo tengo instalado?

Abrir una consola (terminal, línea de comandos, *Power Shell*, *Command Prompt*)

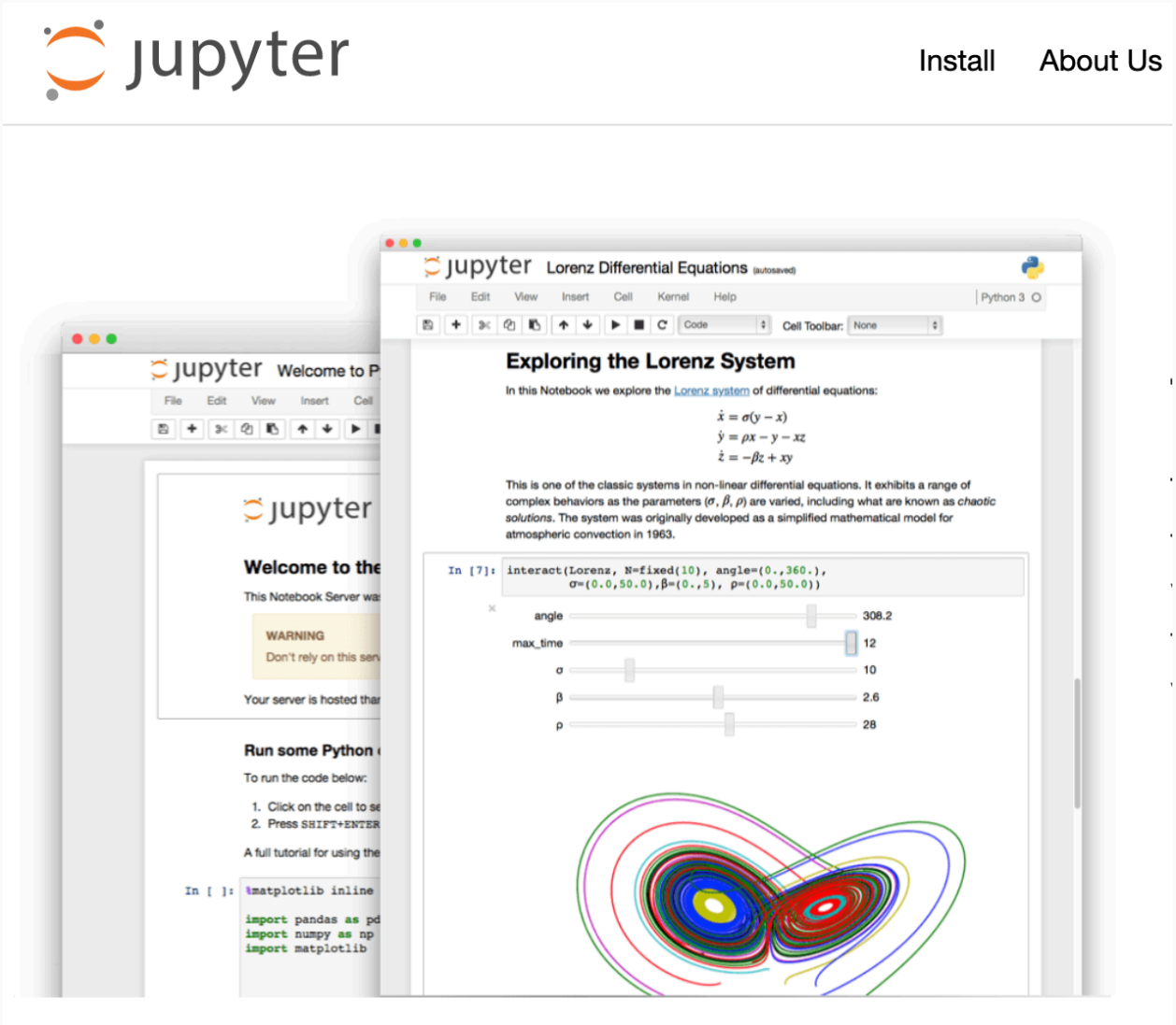
Escribir: `python`, ó `python3`

```
[cruz@LeMacPro:clases$ python
Python 3.7.4 (default, Jul 11 2019, 01:08:00)
[Clang 10.0.1 (clang-1001.0.46.4)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Para salir, escribir: `exit()`

Jupyter-Notebook

Para seguir las clases, leer los apuntes de Raúl, ¡¡y practicar!!



¿Lo tengo instalado?

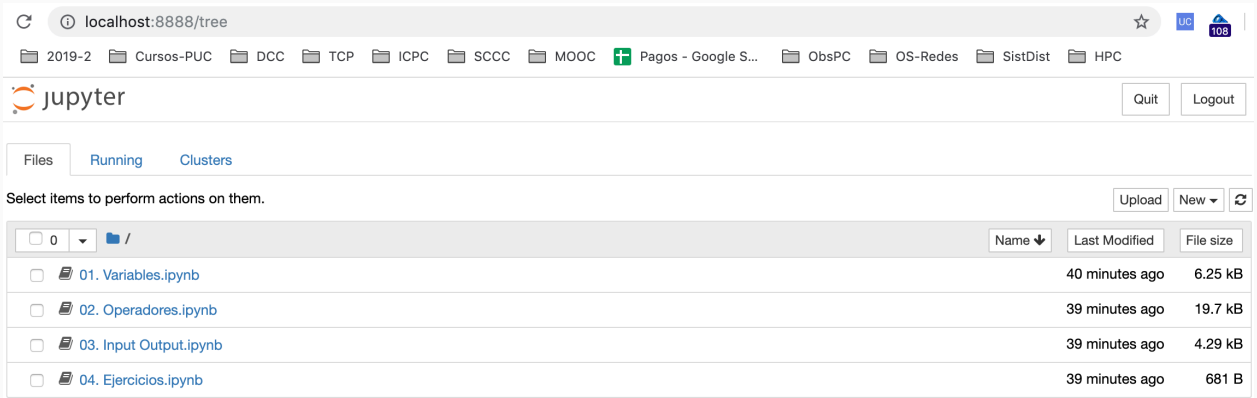
Si tienes instalado Python, probablemente sí

Ejecuta: `jupyter-notebook`

```
cruz@LeMacPro:notebooks-KnowYourselves$ jupyter-notebook
[I 23:05:13.859 NotebookApp] Serving notebooks from local directory: /Users/cruz/PUC/IIC1103-IntroduccionALaProgramacion/2019-2/notebooks-KnowYourselves
[I 23:05:13.859 NotebookApp] The Jupyter Notebook is running at:
[I 23:05:13.859 NotebookApp] http://localhost:8888/?token=1159162400e9bf21fe9fc6e39a2332acb227a81c997f62ab
[I 23:05:13.859 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 23:05:13.880 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///Users/cruz/Library/Jupyter/runtime/nbserver-27731-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=1159162400e9bf21fe9fc6e39a2332acb227a81c997f62ab
```

Se abrirá un navegador (si no lo hace, copia y pega la dirección indicada en la consola)



Si no está instalado: `https://jupyter.org/install`

Contenidos

Actividad Clase 13-Agosto

100 puntos de participación.

Instalar:

- Python 3.7.x
- Jupyter-notebook

Enviar (via SIDING, NO al mail)**:**

1. 50pts. Captura de pantalla luego de ejecutar el archivo `hello.py`
2. 50pts. Captura de pantalla del navegador luego de ejecutar `jupyter notebook` con el archivo `hello.ipynb`
3. Los archivos de apoyo se encuentran en SIDING

Plazo: Miércoles 14-Agosto, 23:59.

Contenidos

Un nuevo lenguaje: Python

Vamos a aprender un nuevo lenguaje: Python

Usaremos este lenguaje para escribir programas, los cuales implementan un algoritmo y podremos ejecutarlos en un computador.

Esto significa que necesitaremos aprender:

- **Sintaxis**: ¿Cómo se escriben frases en este lenguaje?
- **Semántica**: ¿Qué significa cada palabra/frase en este lenguaje?

Un programa en Python

Los programas se escriben como archivo de “texto plano”[1], sin embargo no todo archivo de texto es un programa.

El siguiente es un programa **válido** en Python.

```
a = int(input("Ingrese un número: "))
b = int(input("Ingrese otro número: "))
print("La suma es " + str(a + b))
```

El siguiente **NO** es un programa válido en Python.

```
pide un número
pide otro número
muestra cuánto suman
```

Para ser un programa, el texto debe cumplir con la sintaxis de Python.

[1] texto sin formato

Tipos de datos básicos

Los programas trabajan manipulando datos.

Python es capaz de manejar algunos **tipos de datos básicos**

- Números
  - `int` …`3`, `9`, `65536`, `-2048`
  - `float` …`3.0`, `0.0035`, `65536.5`, `5e6`
- Texto
  - `str` …`"Texto con comillas dobles"` o `'simples'`
- Booleano
  - `bool` …`True`, `False`

Tipos de datos básicos

```
1 5                 #ok
2 3.54              #ok
3 "Hola"            #ok
4 True              #ok
5 true              #fail
```

- El programa es correcto salvo por la línea 6.
- El # es un comentario.

Comentarios: Texto no considerado como parte del código.

Tipos de datos básicos

Comentarios

Texto que es ignorado en la ejecución del programa.

Sirven como ayuda para quien lee el código

```
1  # Comento una línea
2  # Esto no cuenta como
3  # código
4
5  """
6  Comento varias líneas
7  Tampoco cuenta como código
8  """
9
10 esto sí cuenta como código,
11 pero no es un programa válido en Python,
12 por lo tanto no va a funcionar.
```

Operaciones

No es suficiente poder representar datos. Debemos ser capaces de operar sobre los datos.

Las **operaciones** sobre datos permiten obtener nuevos valores

Las operaciones posibles dependen de cada tipo de dato.

- Números: `int`, `float`
- Texto: `str`
- Booleano: `bool`

Operaciones sobre números: aritméticas

```
4 + 2          # Suma                      4+2 => 6
4 - 2          # Resta                     4-2 => 2
-7             # Negación                   -7 => -7
3 * 4          # Multiplicación            3*4 => 12
2 ** 3         # Exponente                 2**3 => 8
3.5 / 2        # División                  3.5/2 => 1.75
3.5 // 2       # División entera           3.5//2 => 1.0
7 % 2          # Módulo (resto de div. entera) 7%2 => 1
```

Prueben también:

- `1.2 - 1.0`
- `0.1 + 0.1 + 0.1`
- `3/10`

Operaciones sobre números: aritméticas

Operador	Descripción	Aridad	Precedencia
**	Exponente	Binario	1
+	Identidad	Unario	2
-	Negación	Unario	2
*	Multiplicación	Binario	3
/	División	Binario	3
//	División entera	Binario	3
%	Módulo	Binario	3
+	Suma	Binario	4
-	Resta	Binario	4

Si no recuerda las precedencias, use paréntesis.

Operaciones sobre números: aritméticas

Ejemplos:

```
3-2+9          # => +10
3-(2+9)        # => -8
1+3*2          # => +7
(1+3)*2        # => +8
-2**2          # => -4
(-2)**2        # => +4
5**3**2        # => ??
```

Exponencial es asociativa por la derecha.

Operaciones sobre números: comparaciones

Los operadores de comparación entregan como resultado un valor de tipo `bool`.

Operador	Descripción
`a == b`	Retorna `True` si y solo si a *es igual a* b
`a != b`	Retorna `True` si y solo si a *es distinto de* b
`a < b`	Retorna `True` si y solo si a *es menor que* b
`a <= b`	Retorna `True` si y solo si a *es menor o igual que* b
`a > b`	Retorna `True` si y solo si a *es mayor que* b
`a >= b`	Retorna `True` si y solo si a *es mayor o igual que* b

Operaciones sobre números: comparaciones

Comparaciones numéricas

```
8 == 8    # >>> True
8 == 9    # >>> False
8 != 9    # >>> True
8 <= 9    # >>> True
8 > 8     # >>> False
```

Todas las operaciones de comparación entregan (**retornan**) un resultado de tipo *boolean*: `True`, ó `False`

Operaciones sobre texto (`str`)

Dos operaciones básicas.

Operador	Descripción
`s1 + s2`	Retorna un `str` que es la **concatenación** de `s1` y `s2`
`s * n`	Retorna un `str` que resulta de concatenar `n` veces el `str` `s`

Operaciones sobre texto (`str`)

`+`: Concatenación de *strings*

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

`*`: Repetición de *strings* (siempre debe haber un entero)

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

Y podemos mezclarlos

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

Operatoria sobre booleanos

Operadores lógicos sobre `bool`.

Operador	Descripción
`not a`	Retorna `True` si y solo sí *a* es `False`
`a or b`	Retorna `True` si y solo sí *a* **ó** *b* son `True` (o ambos)
`a and b`	Retorna `True` si y solo sí *a* **y** *b* son `True`

Operatoria booleana

Ejemplos:

```
1 not True                  # >>> ?
2 True and True             # >>> ?
3 True and False            # >>> ?
4 True or False             # >>> ?
5 False or True             # >>> ?
6 not (False or False)      # >>> ?
```

Operatoria booleana

Ejemplos:

```
1 not True                  # >>> False
2 True and True             # >>> True
3 True and False            # >>> False
4 True or False             # >>> True
5 False or True             # >>> True
6 not (False or False)      # >>> True
```

Operatoria booleana

Ejemplo práctico:

```
1 a = int(input("Ingrese número del 0 al 9: "))
2 condicion = (0 <= a) and (a <= 9)
3 print(condicion)
```

Podemos mezclar **comparaciones** con **operadores booleanos**

Próxima clase

- Almacenar datos: variables, `=`
- Salida de datos: `print`
- Entrada de datos: `input`

Estén atentos a los avisos. Se vienen:

- Instrucciones para crear cuenta *hackerrank* (laboratorios)
- Apuntes complementarios preparados por Raúl
- Actividad de participación para el miércoles 14
- Actividad de participación para el martes 23
- Enlaces útiles

¡Que tengan un buen fin de semana!