



Pontificia Universidad Católica de Chile  
Escuela de Ingeniería  
Departamento de Ciencias de la Computación

## Clase 03: Python y Variables

Rodrigo Toro Icarte (rntoro@uc.cl)

IIC1103 Introducción a la Programación - Sección 5

11 de Marzo, 2015

# Duda vía mail...

¿Qué tiene que ver **Python** con **Sublime Text**?



# Encuesta rápida

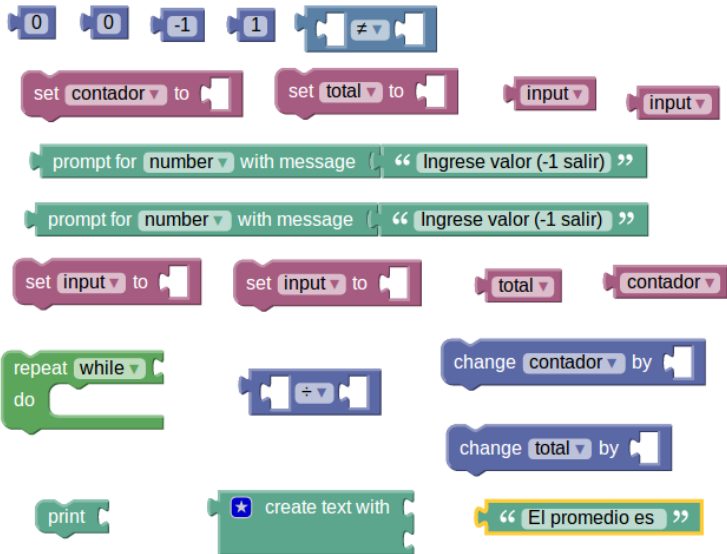
¿Cómo van con los hitos de la semana?

- Instalar Python3 y ejecutar su primer programa.
- Juegos de blockly.
- Ejercicio blockly code.

## Ejercicio 2

**Problema 2:** Cree un programa que calcule el promedio de los números ingresados por el usuario. Mientras no se ingrese un -1, el programa debe seguir pidiendo datos. Cuando se ingrese un -1, el programa muestra el promedio y finaliza. Use los bloques presentes en este **link**.

## Ejercicio 2



## Ejercicio 2 (observaciones)

- Tienen 3 variables:
  - **input**: Para guardar el número ingresado por el usuario.
  - **Total**: Para guardar la suma de los números ingresados.
  - **Contador**: Para guardar la cantidad de números ingresados.

## Ejercicio 2 (observaciones)

- Tienen 3 variables:
  - **input**: Para guardar el número ingresado por el usuario.
  - **Total**: Para guardar la suma de los números ingresados.
  - **Contador**: Para guardar la cantidad de números ingresados.
- **repeat** permite ejecutar código mientras se cumpla cierta condición (bloque puesto a su derecha).
- **change** suma al valor actual de la variable el bloque de la derecha.

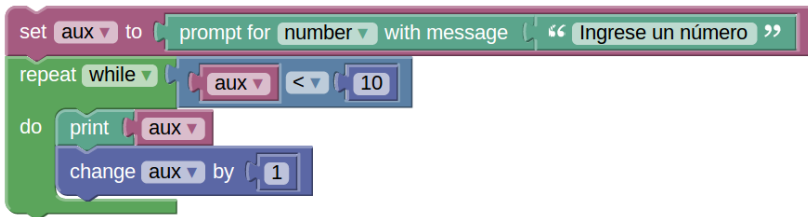
## Ejercicio 2 (observaciones)

- Tienen 3 variables:
  - **input**: Para guardar el número ingresado por el usuario.
  - **Total**: Para guardar la suma de los números ingresados.
  - **Contador**: Para guardar la cantidad de números ingresados.
- **repeat** permite ejecutar código mientras se cumpla cierta condición (bloque puesto a su derecha).
- **change** suma al valor actual de la variable el bloque de la derecha.
- El promedio será **total** dividido por **contador**.



## Ejercicio 2 (observaciones)

Ejemplo ([link](#)):




## Ejercicio 3


*“ programe un marcador de un partido de fútbol. El usuario debe indicar quién anotó un gol (local o visita) o si terminó el partido. Con cada anotación debe actualizarse y mostrarse el marcador. Al finalizar el partido, se debe indicar qué equipo ganó el encuentro.”*

## Ejercicio 3: Ejemplo

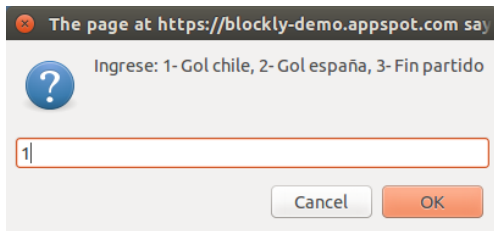
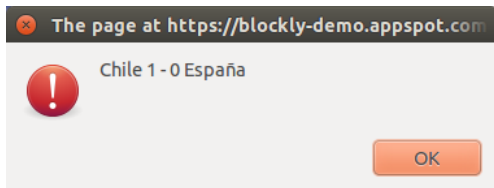
 The page at <https://blockly-demo.appspot.com> says:

 Se viene un partidazo, Chile contra españa, esperemos que estos 90 minutos sean muy emocionantes.

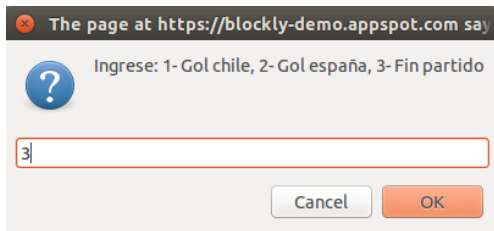
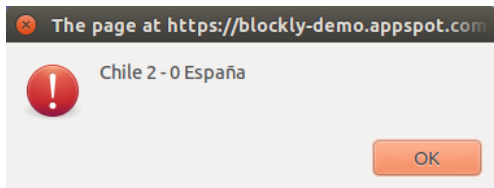
 The page at <https://blockly-demo.appspot.com> say

 Ingrese: 1- Gol chile, 2- Gol españa, 3- Fin partido

## Ejercicio 3: Ejemplo



## Ejercicio 3: Ejemplo



## Ejercicio 3: Ejemplo



Créditos a José Tomás Pérez.

## Ejercicio 3: Observaciones

- ¿Variables?

## Ejercicio 3: Observaciones

- ¿Variables?
  - Input.
  - Chile.
  - España.

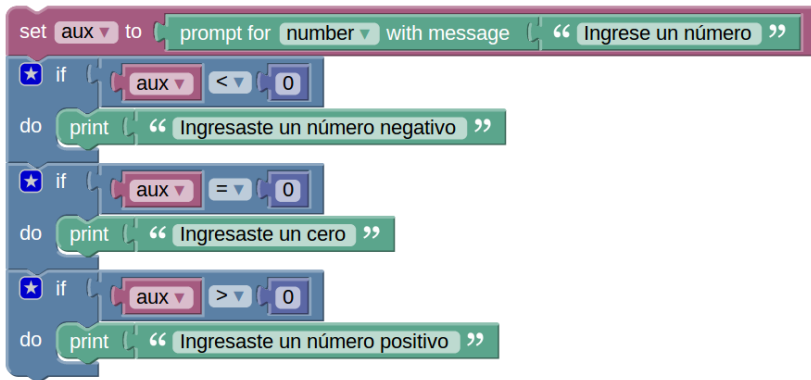


## Ejercicio 3: Observaciones

- ¿Variables?
  - Input.
  - Chile.
  - España.
- Se necesita un **repeat**.
- Se necesita un condicional (**if**).

## Ejercicio 3: Observaciones

Ejemplo ([link](#)):



```
set aux to prompt for number with message " Ingrese un número "
```

```
if aux < 0
```

```
do print " Ingresaste un número negativo "
```

```
if aux = 0
```

```
do print " Ingresaste un cero "
```

```
if aux > 0
```

```
do print " Ingresaste un número positivo "
```

The image shows a Scratch script with the following blocks:

- A **set** block: `set aux to prompt for number with message " Ingrese un número "`
- An **if** block: `if aux < 0`
- A **do** block: `print " Ingresaste un número negativo "`
- An **if** block: `if aux = 0`
- A **do** block: `print " Ingresaste un cero "`
- An **if** block: `if aux > 0`
- A **do** block: `print " Ingresaste un número positivo "`

# Python

Contenidos del curso:

# Python

Contenidos del curso:

- Tipos de datos básicos.
- Control de flujo.
- Funciones.
- String.
- Manejo de Archivos.
- Listas y Tuplas.
- Búsqueda y Ordenamiento.
- Programación Orientada a Objetos.
- Simulación.
- Recursión.

# Python

Contenidos del curso:

- **Tipos de datos básicos.**
- Control de flujo.
- Funciones.
- String.
- Manejo de Archivos.
- Listas y Tuplas.
- Búsqueda y Ordenamiento.
- Programación Orientada a Objetos.
- Simulación.
- Recursión.

# Tipos de datos básicos

## Importante

Todo programa es un archivo de texto, pero no todo archivo de texto es un programa.

# Tipos de datos básicos

## Importante

Todo programa es un archivo de texto, pero no todo archivo de texto es un programa.

## Ejemplos:

```
1 a = int(input("Ingrese un número: "))
2 b = int(input("Ingrese otro número: "))
3 print("La suma es", a + b)
```

```
1 pide un número
2 pide otro número
3 muestra cuánto suman
```

# Tipos de datos básicos

## Importante

Todo programa es un archivo de texto, pero no todo archivo de texto es un programa.

## Ejemplos:

```
1 a = int(input("Ingrese un número: "))
2 b = int(input("Ingrese otro número: "))
3 print("La suma es", a + b)
```

```
1 pide un número
2 pide otro número
3 muestra cuánto suman
```

¿Qué es un programa válido?



# Tipos de datos básicos

Python trabaja con datos.

# Tipos de datos básicos

Python trabaja con datos.

## ① Números

- int (3)
- float (3.0)
- complex (3 + 0j)

## ② Texto

- str (“Texto con comillas dobles” o ‘simples’)

## ③ Booleano

- bool (True, False)

# Tipos de datos básicos

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

# Tipos de datos básicos

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

## Observaciones:

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

# Tipos de datos básicos

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

## Observaciones:

- 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

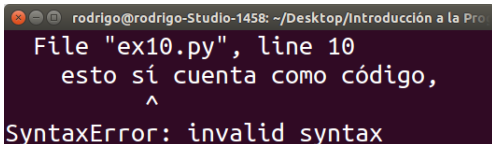
## Comentarios:

- #: Comentar línea
- """ : Comentar trozo de texto.

```
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 por lo tanto no va a funcionar.
```

# Tipos de datos básicos

```
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 por lo tanto no va a funcionar.
```



A screenshot of a terminal window with a dark background. The window title is "rodrigo@rodrigo-Studio-1458: ~/Desktop/Introducción a la Pro". The terminal output shows a Python syntax error: "File 'ex10.py', line 10", "esto sí cuenta como código,", followed by a caret (^) pointing to the space before "por lo tanto" in the previous block's code, and "SyntaxError: invalid syntax".

# Tipos de datos básicos

Operaciones posibles:

- 1 Números
- 2 Texto
- 3 Booleano



# Tipos de datos básicos

Operaciones posibles:

- 1 **Números**
- 2 Texto
- 3 Booleano

# Números: operaciones

1	4+2	# Suma	4+2 => 6
2	4-2	# Resta	4-2 => 2
3	-7	# Negación	-7 => -7
4	3*4	# Multiplicación	3*4 => 12
5	2**3	# Exponente	2**3 => 8
6	3.5/2	# División	3.5/2 => 1.75
7	3.5//2	# División entera	3.5//2 => 1.0
8	7%2	# Módulo	7%2 => 1

# Números: operaciones

1	4+2	# Suma	4+2 => 6
2	4-2	# Resta	4-2 => 2
3	-7	# Negación	-7 => -7
4	3*4	# Multiplicación	3*4 => 12
5	2**3	# Exponente	2**3 => 8
6	3.5/2	# División	3.5/2 => 1.75
7	3.5//2	# División entera	3.5//2 => 1.0
8	7%2	# Módulo	7%2 => 1

**Observación:** recuerden \*\*, // y %.

# Números: operaciones

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

# Números: operaciones

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

**Obs:** Para ahorrarse problemas, usen paréntesis.

# Números: operaciones

## Ejemplos:

1	$3-2+9$	# => +10
2	$3-(2+9)$	# => -8
3	$1+3*2$	# => +7
4	$(1+3)*2$	# => +8
5	$-2**2$	# => -4
6	$(-2)**2$	# => +4
7	$5**3**2$	# => ??

**Observación:** exponencial es asociativa por la derecha.

# Variables

## Motivación

Queremos operar sobre resultados previos... es decir, necesitamos recordar.

# Variables

## Motivación

Queremos operar sobre resultados previos... es decir, necesitamos recordar.

## Definición

Una variable es un nombre simbólico utilizado para acceder a un valor almacenado en la memoria del programa.



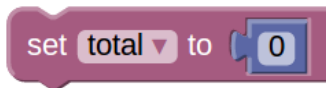
# Variables

## Motivación

Queremos operar sobre resultados previos... es decir, necesitamos recordar.

## Definición

Una variable es un nombre simbólico utilizado para acceder a un valor almacenado en la memoria del programa.



# Variables

## Sintaxis

```
nombre_variable = valor
```

# Variables

## Sintaxis

```
nombre_variable = valor
```

## Asignar

*Dícese del acto de dar valor a una variable.*

# Variables

## Sintaxis

nombre\_variable = valor

## Asignar

*Dícese del acto de dar valor a una variable.*

```
1 a = 3-2+9      # Ahora "a" tiene valor 10
2 b = 4          # Ahora "b" tiene valor 4
3 pi = 3.1415    # Ahora "pi" tiene valor 3.1415
```

# Variables

## Sintaxis

nombre\_variable = valor

## Asignar

*Dícese del acto de dar valor a una variable.*

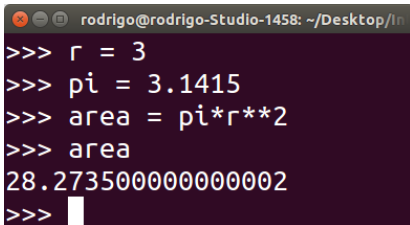
```
1 a = 3-2+9      # Ahora "a" tiene valor 10
2 b = 4          # Ahora "b" tiene valor 4
3 pi = 3.1415    # Ahora "pi" tiene valor 3.1415
```

¿Cuál es la gracia?

# Variables

Ahora podemos operar sobre las variables y ver sus valores.

```
1 r = 3                # Asigno valor de r (radio)
2 pi = 3.1415         # Asigno valor de pi
3 area = pi*r**2     # Calculo área
4 print(area)        # Muestro área
```



A screenshot of a terminal window with a dark background. The window title is "rodrigo@rodrigo-Studio-1458: ~/Desktop/ln". The terminal shows the following Python code being executed line by line:

```
>>> r = 3
>>> pi = 3.1415
>>> area = pi*r**2
>>> area
28.273500000000002
>>>
```

# Variables

Otro ejemplo...

```
1 # Python como calculadora
2 res = 5           # Asigno valor inicial a res
3 res = res*2      # Multiplico res por 2
4 res = res**2     # res elevado a 2
5 res = res%6      # Módulo 5
6 print(res)       # Muestro resultado en consola (4)
```

# Variables

Operaciones del tipo:  $var = var$  (**op**)  $c$  son frecuentes.



# Variables

Operaciones del tipo:  $var = var (op) c$  son frecuentes.

Python permite ahorrar sintaxis usando:  $var (op) = c$

donde  $(op) \in \{+, -, *, **, /, //, \%\}$  y  $c$  es un número.

# Variables

Aprovechando las facilidades que da python.

```
1 # Antes
2 res = 5
3 res = res*2
4 res = res**2
5 res = res%6
6 print(res)
```

```
1 # Después
2 res = 5
3 res *= 2
4 res **= 2
5 res %= 6
6 print(res)
```

# Variables

**Pregunta:** ¿todo nombre es válido para una variable?

---

<sup>1</sup>and, asset, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, field

# Variables

**Pregunta:** ¿todo nombre es válido para una variable?

**Reglas:**

- 1 Debe comenzar con '\_' o una letra (sin ñ ni tildes).
- 2 El resto pueden ser  $\{a - z, A - Z, 0 - 9, _\}$  (sin ñ ni tildes).
- 3 No puede ser una palabra reservada<sup>1</sup>.

---

<sup>1</sup>and, asset, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, field

# Variables

**Pregunta:** ¿todo nombre es válido para una variable?

**Reglas:**

- 1 Debe comenzar con '\_' o una letra (sin ñ ni tildes).
- 2 El resto pueden ser  $\{a - z, A - Z, 0 - 9, _\}$  (sin ñ ni tildes).
- 3 No puede ser una palabra reservada<sup>1</sup>.

**Ejemplo:** casa, casa\_, \_casa, Casa, CaSa01, ...

---

<sup>1</sup>and, asset, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, field

# Variables

**Pregunta:** ¿esto es magia?

---

<sup>2</sup>Pueden ver la dirección en memoria con: *hex(id(nombre\_variable))*.

# Variables

**Pregunta:** ¿esto es magia?

$res \rightarrow 0x282e111$  <sup>2</sup>

Dirección	valor
...	...
0x282e110	<i>algo</i>
0x282e111	4
0x282e112	<i>algo</i>
...	...

Memoria RAM del computador.

---

<sup>2</sup>Pueden ver la dirección en memoria con:  $hex(id(nombre\_variable))$ .

# Variables

**Pregunta:** ¿de qué tipo es la variable x?

---

<sup>3</sup>Pueden ver el tipo de una variable con: `type(nombre_variable)`



# Variables

**Pregunta:** ¿de qué tipo es la variable x?

Depende de la última *asignación* de x.<sup>3</sup>

```
1 x = 3           # x es int
2 x = 3.0        # x ahora es float
3 x = 3 + 0j     # x ahora es complex
4 x = "3"        # x ahora es str
5 x = True       # x ahora es bool
```

---

<sup>3</sup>Pueden ver el tipo de una variable con: `type(nombre_variable)`

# Variables

**Pregunta:** ¿de qué tipo es la variable x?

Depende de la última *asignación* de x.<sup>3</sup>

```
1 x = 3          # x es int
2 x = 3.0       # x ahora es float
3 x = 3 + 0j    # x ahora es complex
4 x = "3"       # x ahora es str
5 x = True      # x ahora es bool
```

¿Esto importa?

---

<sup>3</sup>Pueden ver el tipo de una variable con: `type(nombre_variable)`

# Variables

**Pregunta:** ¿de qué tipo es la variable x?

Depende de la última *asignación* de x.<sup>3</sup>

```
1 x = 3          # x es int
2 x = 3.0       # x ahora es float
3 x = 3 + 0j    # x ahora es complex
4 x = "3"       # x ahora es str
5 x = True      # x ahora es bool
```

¿Esto importa? → Sí!

Ej: ¿Cuánto es  $3^*x$ ?

---

<sup>3</sup>Pueden ver el tipo de una variable con: `type(nombre_variable)`

# Variables

**Pregunta:** ¿Cómo cambio el tipo de la variable?

# Variables

**Pregunta:** ¿Cómo cambio el tipo de la variable?

Para *castear* una variable 'x' se usa: `int(x)`, `float(x)`, `complex(x)`, `str(x)` y `bool(x)`.

```
1 x = 3 # x es int de valor 3
2 x = float(x) # x ahora es float de valor 3.0
3 x = complex(x) # x ahora es complex de valor 3 + 0j
4 x = str(x) # x ahora es str de valor "3"
5 x = bool(x) # x ahora es bool de valor True
6 x = int(x) # x vuelve a ser int de valor 1
```

# Variables

**Pregunta:** ¿Cómo cambio el tipo de la variable?

Para *castear* una variable 'x' se usa: `int(x)`, `float(x)`, `complex(x)`, `str(x)` y `bool(x)`.

```
1 x = 3 # x es int de valor 3
2 x = float(x) # x ahora es float de valor 3.0
3 x = complex(x) # x ahora es complex de valor 3 + 0j
4 x = str(x) # x ahora es str de valor "3"
5 x = bool(x) # x ahora es bool de valor True
6 x = int(x) # x vuelve a ser int de valor 1
```

**Obs:** `bool(x)` es `True` ssi  $x \neq 0$ .

# Ejercicios

- 1) Evalúe polinomio  $x^4 + \frac{1}{2}x^3 + 2x^2 - x$  para un  $x$  cualquiera.
- 2) Obtenga la unidad de una variable  $x$ . (ej: si  $x = 123$ , debe obtener 3).
- 3) Obtenga la decena de una variable  $x$ . (ej: si  $x = 123$ , debe obtener 2).