Piezas de un lenguaje de programación

Introducción a la Programación

Departamento de Ciencias Básicas, UNLu

Elementos básicos para construir programas (en cualquier lenguaje de programación!)

- **Instrucciones** Le dicen a la computadora de hacer algo en una línea de código
- **Tipo de dato** Los datos tienen diferentes tipos
- **Variables** Nos permiten almacenar datos
- **Operadores** Sirven para manipular datos
- **Condicionales** Ejecuta un bloque de código si una condición es satisfecha
- **Funciones** Los programas son estructurados mediante funciones.
- **Ciclos** Ejecuta un bloque de código múltiples veces

Elementos básicos para construir programas: Instrucciones

Las instrucciones son las órdenes que entiende el lenguaje de programación. En general cada línea de un programa Python corresponde a una instrucción.

Algunos ejemplos de instrucciones:

```
print("Hello world!")
Hello world!

print(911)
911
```

Elementos básicos para construir programas:
Tipos de dato

Un tipo abstracto de datos (TAD) es un modelo matemático compuesto por una colección de operaciones definidas sobre un conjunto de datos.

Ejemplo de TAD: El conjunto de los números enteros ..., −2, −1, 0, 1, 2, ..., y las operaciones permitidas para los números enteros: suma, resta, multiplicación, división, etc. Un TAD es un modelo matemático abstracto, y es independiente de cómo la computadora represente los enteros.

Elementos básicos para construir programas:
Tipos de dato

Un tipo de dato es la implementación de un TAD en algún lenguaje de programación.

Tipo de dato (Python)	Descripción	Ejemplos
int (integer)	valores enteros	3, 8, 999
float	fracciones	3.1416
str (string)	texto	“Hello World!”
bool (boolean)	valores booleanos	True, False
NoneType	ausencia de valor	None

Elementos básicos para construir programas:
Variables

Las variables permiten asignarle un nombre a un valor, de tal forma de "recordarlo" para usarlo posteriormente.

```
nombre_de_variable = valor
```

Ejemplo 1: Se crea una variable llamada 'nombre_materia' y le asignamos el valor "Introducción a la Programación" que es de tipo **`string`**:

```
nombre_materia = "Introducción a la Programación"
```

Ejemplo 2: Se crea una variable llamada 'codigo_materia' y le asignamos el valor 11071 que es de tipo **`integer`**:

```
codigo_materia = 11071
```

Al ejecutar la instrucción de asignación, la variable se crea y su tipo de dato es del valor asignado.

Elementos básicos para construir programas:
Variables y tipos de dato

En Python, si a una variable se le asigna otro valor de otro tipo, la variable cambia de tipo. En otros lenguajes de programación, el tipo de dato de una variable una vez creada es fijo.

A la variable 'x' le asignamos el valor "Hello World":

```
x = "Hello World"
```

La variable 'x' ahora es de tipo **`string`**.

A la variable 'x' le asignamos el valor 45:

```
x = 45
```

La variable 'x' ahora es de tipo **`integer`**.

Elementos básicos para construir programas:
Variables y tipos de dato

En ciencias de la computación la conversión de tipos (type casting en inglés) se refiere a la transformación de un tipo de dato en otro.

A la variable 'x' le asignamos el valor '45':

```
x = '45'
```

La variable 'x' ahora es de tipo **`string`**.

Si hacemos casting:

```
x = int('45')
```

La variable 'x' ahora es de tipo **`integer`**.

Elementos básicos para construir programas:
Averiguando el tipo de dato de una variable

```
rosana@cerebro:~$ python3
Python 3.6.10 |Anaconda, Inc.| (default, Jan  7 2020, 21:14:29)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x=3
>>> type(x)
<class 'int'>
>>> x=3.3
>>> type(x)
<class 'float'>
>>> x='3.3'
>>> type(x)
<class 'str'>
>>> x=float('3.3')
>>> type(x)
<class 'float'>
>>> x=True
>>> type(x)
<class 'bool'>
>>> x=None
>>> type(x)
<class 'NoneType'>
```

Elementos básicos para construir programas:
Operadores

Los operadores sirven para manipular datos. Un operador tiene cierta tipo y tiene un orden de precedencia con respecto a los otros operadores.

Ejemplos:

```
x = 3 + 5
print("Hello" + " World")
print(1.5 + 1.5)
x = 2*x + 2**3
print(x > 25)
print((x < 25) and (x >= 0))
```

Elementos básicos para construir programas:
Tipos de Operadores

- Operadores aritméticos: +, -, *, /, //, **, %
- Operadores de asignación: +=, -=, *=, /=, //=, %=
- Operadores de comparación: ==, !=, <, <=, >, >=
- Operadores booleanos: not, or, and

Elementos básicos para construir programas:
Precedencia de Operadores

1. Exponenciación: **
2. Multiplicación, división, módulo: *, /, //, %
3. Suma y resta: +, -
4. Operadores de comparación: ==, !=, <, <=, >, >=
5. Operador booleano de negación: not
6. Operadores booleanos and, or

La precedencia de los operadores se puede modificar mediante paréntesis.

Ejemplo:
`2+3**2` dará como resultado `11`.
`(2+3)**2` dará como resultado `25`.

Elementos básicos para construir programas:
Condicionales

Un condicional ejecuta un bloque de código si una condición es satisfecha.

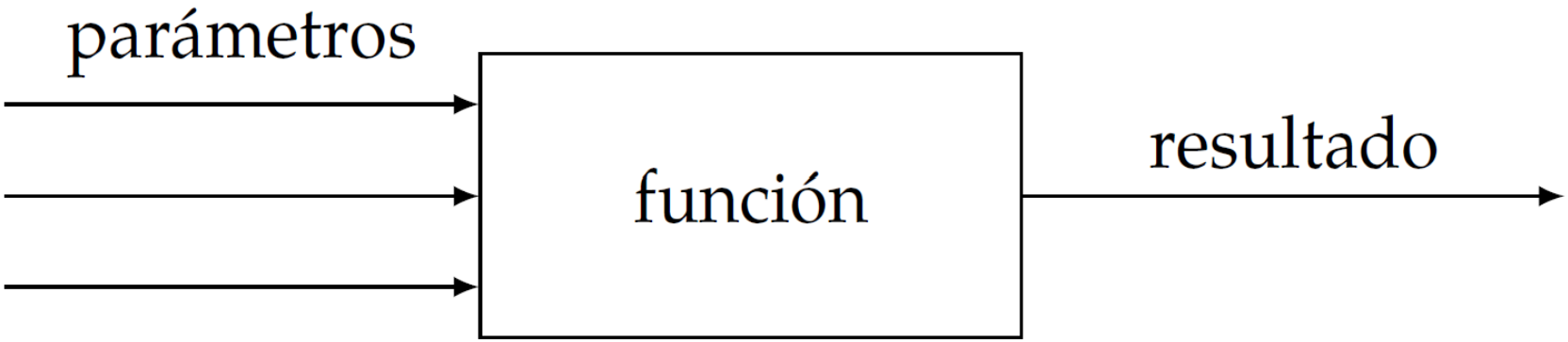
Ejemplo:

```
if 3 > 1:
  print("Hello World!")
Hello World!
```

Elementos básicos para construir programas: Funciones

Una función es un fragmento de programa que permite efectuar una operación determinada.

Una función puede recibir cero o más parámetros o argumentos (expresados entre entre paréntesis, y separados por comas), efectúa una operación y devuelve un resultado.



Elementos básicos para construir programas: Funciones

Las funciones en Python las definimos así:

```
def cuadrado(x):
  y = x*x
  return y
```

`def cuadrado(x):` le indica a Phython que estamos escribiendo una función cuyo nombre es cuadrado y que debe recibir 1 parámetro. La instrucción **`return`** indica cuál es el resultado de la función.

Ejemplo: para calcular el cuadrado de 3, la invocamos así

```
cuadrado(3)
```

Elementos básicos para construir programas:
Ciclos

Los ciclos nos permiten repetir secuencias de instrucciones.

Un ciclo es definido, si ya sé de antemano cuántas veces repetiré las instrucciones del ciclo. La instrucción que determina cuántas veces se realizará el ciclo es el encabezado del ciclo, y las instrucciones que describen la acción que se repite componen el cuerpo del ciclo.

Elementos básicos para construir programas:
Ciclos

Un ciclo definido en Python es de la forma:

```
for <nombre> in <expresión>:
  <cuerpo>
```

Una vez evaluada la <expresión> (cuyo resultado debe ser una *secuencia de valores*), se sabe exactamente cuántas veces se ejecutará el <cuerpo> y qué valores tomará la variable <nombre>.

Elementos básicos para construir programas:
Ciclos

Ejemplo de ciclo definido en Python:

```
for x in range(1, 3):
  print("Hello world")
```

`range(numero1, numero2)` genera la secuencia de valores en el intervalo de enteros `[numero1, numero2-1]`. Entonces, `range(1, 3)` genera la secuencia `[1, 2]`, y el ciclo hará 2 iteraciones. En la primera iteración del ciclo, x toma el valor 1, y se imprime "Hello world" por pantalla. En la segunda iteración, x toma el valor 2, y se imprime en la pantalla nuevamente "Hello world".

Expresiones

Una expresión es una porción de código que produce o calcula un valor.

Una expresión puede ser:

- Un valor literal. Por ejemplo, la expresión 33, produce el valor numérico 33.
- Una variable. El valor que produce es el que tiene guardado la variable.
- Los operadores permiten combinar expresiones y construir expresiones más complejas.
- Una llamada a función.