



UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
Fundamentos de Computación y Programación (10110-1)



CLASE N°4

USANDO FUNCIONES EN PYTHON

Funciones



- Python provee **funciones primitivas** de las cuáles hoy aprendimos cinco:
 - **Valor absoluto** y **potencia** y las operaciones para cambiar el tipo a **entero**, **entero largo** y **número de punto flotante**
- Las funciones en Python siguen el concepto matemático de función:
 - Tienen un **nombre**
 - Se le entregan **argumentos** sobre los cuales se aplica
 - **Siempre devuelven un valor (MEDIANTE RETURN)**

2

Funciones importadas



- Podemos extender la funcionalidad de Python **importando funciones**
 - Desde bibliotecas llamadas **módulos**
 - Hay dos formas para importarlas/usarlas:

Forma de importar	Forma de usar
import <nombre módulo> Ej: <code>import math</code>	<nombre modulo>.<nombre función>(parámetros) Ej: <code>math.sqrt(2)</code>
from <nombre módulo> import <nombre función> Ej: <code>from math import sqrt</code>	<nombre función>(parámetros) Ej: <code>sqrt(2)</code>

3

Funciones importadas: Módulo math




- El módulo **math** contiene varias funciones matemáticas y trigonométricas, por ejemplo:

Nombre	Descripción
sin(x)	Seno de x, con x expresado en radianes
cos(x)	Coseno de x, con x expresado en radianes
tan(x)	Tangente de x, con x expresado en radianes
exp(x)	Número <i>e</i> elevado a x
log(x)	Logaritmo natural (base <i>e</i>) de x
log10(x)	Logaritmo en base decimal de x
sqrt(x)	Raíz cuadrada de x
degrees(x)	Convierte a grados un ángulo x expresado en radianes
radians(x)	Convierte a radianes un ángulo x expresado en grados.

4

Funciones propias



- Python también nos permite definir **funciones propias**


```
def sumaCuadrados(x, y):  
    cuadradoX = x * x  
    cuadradoY = y * y  
    resultado = cuadradoX + cuadradoY  
    return resultado
```

Diagram illustrating the structure of a Python function definition:

- Encabezado** (Header): `def sumaCuadrados(x, y):`
 - Nombre de la función** (Function name): `sumaCuadrados`
 - Parámetros formales** (Formal parameters): `(x, y)`
- Cuerpo** (Body):
 - `cadradoX = x * x`
 - `cadradoY = y * y`
 - `resultado = cuadradoX + cuadradoY`
 - `return resultado`
- Retorno** (Return): `return`
- Línea en blanco** (Blank line): Indicated by a bracket below the `return` statement.
- Indentación** (Indentation): Indicated by a bracket for the body lines.

5

Parámetros formales y reales



- Recordemos:
 - Al declarar una función, indicamos los **parámetros formales** en el encabezado, y los usamos en el cuerpo de la función
 - Al **invocar** la función, lo hacemos con su nombre y especificamos los **parámetros actuales**
 - Al momento de la ejecución, se **asocia** cada parámetro formal a un parámetro actual, de acuerdo a sus **posiciones**
 - Por esto, el número de parámetros actuales a especificar en la invocación, **debe ser igual** al número de parámetros formales declarados para la función

6

Parámetros formales y reales



- En el ejemplo:
 - Se declara la función **sumaCuadrados**
 - Que tiene dos **parámetros formales**: **x** e **y**
 - Podemos invocarla, por ejemplo, con la expresión **sumaCuadrados(2, 3)**
 - El **parámetro formal x** toma el **valor 2** y el **parámetro formal y** toma el **valor 3**
 - Se ejecuta **cada sentencia** en **el orden que aparecen**:

```
cuadradoX = 2 * 2 = 4
cuadradoY = 3 * 3 = 9
resultado = 4 + 9 = 13
return 13
```

7

Variables locales y globales



- Las variables tienen cierta *visibilidad*
 - Técnicamente, a esta visibilidad se le llama **alcance**
- Hay esencialmente dos tipos de alcances:
 - las **variables locales** sólo son visibles en el cuerpo de la función donde se crean (con una sentencia de asignación)
 - las **variables globales** son visibles por todas las sentencias que siguen a su creación, excepto donde existe una variable local con el mismo nombre
- Si una variable es local o global, depende de **dónde es creada**

8


Variables locales y globales



- Las variables globales se crean **fuera** de cualquier función, mientras que las locales **dentro** de ellas.
- Los **parámetros formales** son, esencialmente, **variables locales**

```
>>> valorInicial = 2
>>> PI = 3.1415926535
>>> def calculaInicio(v):
>>>     valorInicial = v ** 2
>>>     return 2 * valorInicial


>>> def calculaFin(v)
>>>     return PI * valorInicial * v
```



Variable local Variable global

9

Para la próxima clase



- Probar **funciones nativas** y del módulo **math**
- Revisar guía de **ejercicios propuestos**
- Comenzaremos a usar distintas funciones importadas y propias para unirlas en un **programa**

10

