FUNDAMENTOS DE ALGORITMOS

MÓDULO 1

RECURSO DE APRENDIZAJE 3

FUNCIONES INTEGRADAS & BIBLIOTECAS DE PYTHON

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Escuela de Ingeniería Informática

FUNCIONES INTEGRADAS

Funciones Integradas

- El **intérprete** de **Python** tiene una serie de funciones y tipos incluidos que están **siempre disponibles**.

- **Las funciones integradas de Python en orden alfabético son:**

  - **abs**, all, any, ascii, bin, bool, breakpoint, bytearray, bytes, callable, chr, classmethod, compile, complex, delattr, dict, dir, divmod, enumerate, eval, exec, filter, **float**, format, frozenset, getattr, globals, hasattr, hash, help, hex, id, **input**, **int**, isinstance, issubclass, iter, len, list, locals, map, max, memoryview, min, next, object, oct, open, ord, **pow**, **print**, property, range, repr, reversed, **round**, set, setattr, slice, sorted, staticmethod, **str**, sum, super, tuple, type, vars, zip, __import__

FUNCIONES ABS, POW y ROUND

Función abs()

- Esta función retorna el **valor absoluto** de un **número** traspasado como **argumento.**

Sintaxis de llamada : **abs**(**número**)

- El **argumento** puede ser un **número entero** o de **punto flotante.** Si el **argumento** es un **número complejo,** retorna su **magnitud.**

Llamada función	Valor retornado
abs(-3)	3
abs(3)	3
abs(-3.898)	3.898
abs(3.898)	3.898

Función pow()

- Esta función retorna una **base** elevada a un **exponente.**
- Si el argumento **mod** está presente, retorna **base** elevado a **exponente** módulo **mod**, calculado de manera más eficiente que **pow(base, exp) % mod**

Sintaxis de llamada : **pow**(base, exponente [, mod])

- La forma con dos argumentos **pow(base, exp)** es **equivalente** a usar el **operador** potencia: **base**exp**.**
- Los argumentos deben ser de **tipo numérico.**
- Si **mod** esta presente debe ser de **tipo entero y distinto de cero.**

Ejemplos Uso Función pow()

Llamada función	Valor retornado
pow(2,3)	8
pow(2,0)	1
pow(2,-3)	0.125
pow(2.0,3)	8.0
pow(2.0,0)	1.0
pow(2.0,-3)	0.125
pow(2,0.5)	1.4142135623730951
pow(-1,0.5)	(6.123233995736766e-17+1j)
pow(2,3,7)	1

Función round()

- **Esta función retorna** un número **redondeado** a ndigitos de precisión después del punto decimal.
- Si ndigitos es omitido o es **None**, retorna el entero más cercano.

Sintaxis de llamada : **round**(número [, ndigitos])

- Para los **tipos integrados** que soportan **round**(), los valores son redondeados al múltiplo de 10 más cercano a la potencia menos ndigitos; si dos múltiplos están igual de cerca, el redondeo se hace hacia **la opción par.**

- Así que por ejemplo tanto round(0.5) como round(-0.5) son 0, y round(1.5) es 2.

Ejemplos Uso Función round()

Llamada función	Valor retornado
round(0.5)	0
round(-0.5)	0
round(1.5)	2
round(-1.5)	-2
round(1.57462,2)	1.57
round(1.5756,2)	1.58
round(1.56562,2)	1.57
round(1.56462,2)	1.56
round(2.675, 2)	2.67
round(2.676, 2)	2.68

BIBLIOTECAS DE PYTHON

Bibliotecas de Python

- La **biblioteca** estándar de **Python** ofrece una amplia gama de **funcionalidades.**

- Esta biblioteca contiene :

  - **Módulos** integrados en **lenguaje C** que brindan acceso a **funcionalidades** del sistema, como E/S de archivos por ejemplo.

  - **Módulos** escritos en **Python** que brindan **soluciones estandarizadas** para diferentes tipos de problemas.

MÓDULO MATH

Módulo Math

- Este **módulo** proporciona acceso a las **funciones matemáticas** y **constantes** definidas en el estándar de C.

- Las **funciones** NO pueden ser usadas con **números complejos**. Se deben usar las **funciones** con el **mismo nombre** del módulo **cmath** si se requiere soporte para **números complejos**.

- Excepto cuando se indique lo contrario explícitamente, **todos los valores retornados** son **flotantes**.

Funciones del Módulo Math

Las funciones que incluye el módulo **math** se pueden clasificar en la siguiente categorías :

1. Funciones de **teoría de números** y funciones de **representación**.
2. Funciones **logarítmicas** y **exponenciales**.
3. Funciones **trigonométricas**.
4. Conversión **angular**.
5. Funciones **hiperbólicas**.
6. Funciones **especiales**.

Funciones de teoría de números y representación

math.ceil(x)	math.comb(n, k)	math.copysign(x, y)
math.fabs(x)	math.factorial(x)	math.floor(x)
math.fmod(x, y)	math.frexp(x)	math.fsum(iterable)
math.gcd(a, b)	math.isfinite(x)	math.isinf(x)
math.isnan(x)	math.sqrt(x)	math.ldexp(x, i)
math.modf(x)	math.perm(n, k=None)	math.prod(iterable, start=1)
math.remainder(x, y)	math.trunc(x)	math.isclose(a, b, rel_tol=1e-09, abs_tol=0.0)

- Presentes en la versión 3.8
- Presentes en la versión 3.7

Funciones logarítmicas y exponenciales

`math.exp(x)`	`math.expm1(x)`
`math.log(x[, base])`	`math.log1p(x)`
`math.log2(x)`	`math.log10(x)`
`math.pow(x, y)`	`math.sqrt(x)`

Funciones Trigonométricas

`math.acos(x)`	`math.asin(x)`	`math.hypot(x1,x2,…)`
`math.atan(x)`	`math.atan2(y, x)`	`math.sin(x)`
`math.cos(x)`	`math.dist(p, q)`	`math.tan(x)`

- Presentes en la versión 3.8

Conversión Angular

`math.degrees(x)`	`math.radians(x)`

Funciones Hiperbólicas

`math.acosh(x)`	`math.asinh(x)`	`math.atanh(x)`
`math.cosh(x)`	`math.sinh(x)`	`math.tanh(x)`

Funciones Especiales

`math.erf(x)`	`math.erfc(x)`
`math.gamma(x)`	`math.lgamma(x)`

Constantes del Módulo math

Constante	Descripción
math.pi	La constante matemática π = 3.141592…, hasta la precisión disponible.
math.e	La constante matemática e = 2.718281…, hasta la precisión disponible.
math.tau	La constante matemática τ = 6.283185…, hasta la precisión disponible.
math.inf	Un valor infinito positivo en punto flotante.
math.nan	Un valor de punto flotante que «no es un número» (NaN).

USO DE MÓDULOS

Uso de Módulos (1/2)

- Para poder usar en nuestros **programas** las **funciones** y **constantes** definidas en los **módulos** incluidos en la **biblioteca** de **Python** debemos importarlos con la **sentencia import**.

Sintaxis : **import** nombreModulo

- Para usar una **función/constante** del **módulo importado** debemos indicar el **nombre del módulo**, un **punto** y el **nombre de la función o constante.**

Ejemplo:

```
import math
x = math.sqrt(2) * math.pi
```

Uso de Módulos (2/2)

- Si queremos importar algunas **funciones o constante específicas** de un **módulo,** debemos usar la siguiente **sentencia:**

Sintaxis : `from nombreModulo import nombreFunción, …`

Ejemplo:

```
from math import sqrt , pi
x = sqrt(2) * pi
```