Python Programming

Módulo 1

Funciones built-in (integradas)

Funciones built-in

El intérprete de Python tiene un número de **funciones integradas** *(built-in),* las cuales están siempre disponibles. Algunas las usamos cotidianamente programando en Python, como, por ejemplo:

```
>>> print()
```

Una de las acciones básicas e imprescindibles que tiene que realizar un programa es la de mostrar información por pantalla: texto, números, resultado. La función `print()` es la que nos permite mostrar por pantalla.

```
>>> input()
```

La función `input()` permite a los usuarios introducir datos desde la entrada estándar (normalmente se corresponde con la entrada de un teclado). Los datos son tomados como ‘str’ , aunque los ingresos fueran números, son tomados como string, luego hay que convertirlos.

```
>>> range()
```

La “función” `range()` es algo un poco más complejo que una función, pero se utiliza como si fuera una función.

El `range()` con un único argumento se escribe `range(n)` y crea una secuencia inmutable de n números enteros consecutivos que empiezan en `0` y acaba en `n - 1`.

Con dos argumentos se escribe `range(m, n)` y crea una secuencia inmutable de enteros consecutivos que empieza en `m` y acaba en `n - 1`.

Con tres argumentos se escribe `range(m, n, p)` y crea una secuencia inmutable de enteros que empieza en `m` y acaba `n-1`, aumentando los valores de `p` en `p`. Si `p` es negativo, los valores van disminuyendo de `p` en `p`.

El `range()` es fácil de iterar o recorrer con un `for` y también sirve para crear listas con la función incorporada `list()`.

```
>>> int()
>>> float()
>>> str()
```

Hacer un *cast* o *casting* significa convertir un tipo de dato a otro. Anteriormente hemos visto tipos como los int, string o float. Pues bien, es posible convertir de un tipo a otro con las funciones correspondientes, siempre y cuando se pueda realizar la conversión.

Inclusive se puede convertir diferentes conjuntos a listas, con `list()`, o a tuplas con `tuple()`.

```
>>> len()
```

Retorna el tamaño (el número de elementos) de un objeto. El argumento puede ser una secuencia (como una cadena, un objeto byte, una tupla, lista o un rango) o una colección (como un diccionario, un set o un frozen set).

> Para ver más funciones built-in podés ver la documentación oficial de funciones built-in

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

¡Sigamos trabajando!