Cheatography

Python's "Beautiful heart" Cheat Sheet

by haunter_3 via cheatography.com/208686/cs/44764/

Pieza #4: Operaciones lógicas

Operador de Identidad	Operadores de Comparación	Operador de Membresía	Operadores Lógicos
EL propósito del operador de identidad is ver si dos referr-encias de objetos se refieren al mismo objeto o ver si un objeto es "ninguno"	Estos operadores comparan valores de objetos.	Se utiliza para comprobar la pertenencia de un valor en los tipos de datos de colecciones	Python ofrece tres operadores lógicos: and, or, y not. Tanto and como or utilizan lógica de corto circuito y devuelven el operando que determinó el resultado; no devuelven un valor booleano

>>> a = ["Atencion", 3, None]
>>> b = ["Atencion", 3, None]
>>> a is b
False
	>>> a = 2
	>>> b = 6
	>>> a == b
	False
	>>> a < b
	True
		>>> p = (4, "rana", 9, -33, 9, 2)
		>>> 2 in p
		True
		>>> "dog" not in p
		True
			>>> cinco = 5
			>>> dos = 2
			>>> cero = 0
			>>> cinco and dos
			2
			>>> dos and cinco
			5

Uno de las características fundamentales de cualquier lenguaje de programación son sus operaciones lógicas.

Pieza #5: Declaraciones de Control de flujo

Delcaración *If*	Delcaración *While*	Delcaración *For...in*	Manejo básico de excepciones
La sintáxis general para la declaración *if* de Python es:	Se utiliza para ejecutar una serie de funciones cero o más veces, la cantidfad de veces depende del estado de la expresión booleana del bucle while.	El bucle *for* de Python reutiliza la palabra clave *in* y tiene la siguiente sintáxis:	Una excepción es un objeto como cualquier otro de Python y cuando se convierte en una cadena (por ejemplo, cuando se imprime) la excepción produce un texto de mensaje.

if *expresion_booleana1*:
suite1
elif *expresion_booleana2*:
suite2
elif *expresion_booleanaN*:
suiteN
else:
else_suite
	while *expresion_booleana*:
	suite
		for *variable* in *iterable*:
		suite
			try:
			try_suite
			except *excepcion1* as *variable1*:
			excepcion_suite1
			except *excepcionN* as *variableN*:
			excepcion_suiteN

El lenguaje de Python, un bloque de código, es decir, una secuencia de una o más sentencias, se denomina *suite*.

Pieza #1: Tipos de datos

Python representa números entreros (positivos o negrativos) utilizando el tipo *int* y representa cadenas utilizando *str*
Aquí algunos ejemplos:
int:
125
-315
str:
"Jose Luis"
'Folio 234'

En Python, tanto *str* como los tipos numéricos básicos como *int* son inmuta-

Pieza #6: Operadores Aritméticos

bles, es decir, una vez configurados su valor no se puede cambiar

Pieza #7: Input/Output

Para poder escribir programas realmente útiles, debemos ser capaces de leer la entrada (por ejemplo, del usuario en la consola y de los archivos) y producir una salida, ya sea en la consola o en archivos. Python proporciona la función *input()* incorporada para aceptar la entrada del usuario. Esta función toma un argumento de cadena opcional (que imprime en la consola); luego espera a que el usuario escriba una respuesta y termine presionando Enter (o Return). Si el usuario no escribe ningún texto sino que simplemente presiona Enter, la función *input()* devuelve una cadena vacía; de lo contrario, devuelve una cadena que contiene lo que el usuario escribió, sin ningún terminador de línea..

Pieza #3: Tipos de datos de colección

Python ofrece varios tipos de datos de colección que pueden contener elementos, incluidas matrices y conjuntos asociativos, pero aqui presentaremos solo dos: tuplas y listas.
Las tuplas son creadas usando comas:
('Dinamarca', 'Finlandia', 'Noruega', 'Suiza')
Una forma de crear una lista es utilizar corchetes:
[1, 2, 3, 4, 5]

Todos los elementos de datos de Python son objetos y se utiliza el operador punto ("atributo de acceso") para acceder a lso atributos de un objeto

Python proporciona un conjunto completo de operadores aritméticos, incluidos los operadores binarios para las cuatro operaciones matemáticas básicas:
+ suma
- resta
* multiplicación
/ división
Además, muchos tipos de datos de Python se puden utilizar con operadores de asignación aumentada como += y *=

```
>>> 5 + 6
11
>>> 3 - 7
-4
>>> 4 * 8
32
```

Pieza #2: Referencia de objetos

Una vez que tenemos algunos tipos de datos los siguiente que necesitamos son variables en las que almacenarlos.
La sintáxis es simple *ObjetodeReferencia = valor*
Ejemplo:
x = 25
y = "Blue"

La función *type ()* regresa el tipo de dato del elemento de datos que se le proporciona.

>>> *type(x)* #Imprime 'int'

Pieza #8: Creando y Llamando funciones

Python proporciona un medio para encapsular conjuntos como funciones las cuales pueden ser parametrizdas mediante los argumerntos que se les pasan. La sintáxis general para crear una función es:

def *nombre_funcion* (*argumentos*):
suite

Los argumentos son opcionales y los argumentos múltiples deben estar separados por comas. cada función de Python tiene un valor de retorno; el valor predeterminado es *None*.