Guía rápida de Python – "Cheat sheet"

Lo básico

Los espacios en blanco importan. Tu código no se ejecutará correctamente si utilizas una *indentación* errónea.

```
# Esto es un comentario
```

Lógica básica

Condicional "if"

```
if test:
    # realizar acción si test es True
elif test2:
    # realizar acción si test2 es True
else:
    # realizar acción si las anteriores
    # son falsas (test y test2)
```

Bucle "while"

```
while test:
    # realizar acción mientras test
    # sea True
```

Bucle "for"

```
for item in secuencia:
    # realizar acción para cada miembro
    # (item) de secuencia. Por ejemplo,
    # cada elemento de una lista, o
    # cada caracter de una cadena.

for i in range(10):
    # realizar acción 10 veces (0 a 9)

for i in range(5, 10):
    # realizar acción 5 veces (5 a 9)
```

Cadenas

Una cadena es una secuencia de caracteres, generalmente utilizada para almacenar texto.

Creación

```
cadena = "Hola mundo."
cadena = 'Hola mundo.'
```

Acceso

```
cadena[4]
```

retorna `'a'`

División

```
cadena.split(' ')
cadena.split('n')
```

retorna `['Hola', 'mundo.']`
retorna `['Hola mu', 'do.']`

Para unir una lista de cadenas, utiliza la función `join()` como método de la cadena que utilizarás para separar cada uno de los elementos de la lista (o bien `''`).

```
palabras = ['Esto', 'es', 'una', 'lista', 'de',
            'cadenas']

' '.join(palabras
'ZOOL'.join(palabras)
''.join(palabras)
```

retorna `"Esto es una lista de cadenas"`
retorna `"EstoZOOLesZOOLunaZOOLlistaZOOLdeZOOLcadenas"`
retorna `"Estoesunalistadecadenas"`

Formateo de cadenas: similar a la función de C `printf()`, utiliza en su lugar el operador `%` para añadir los elementos de una tupla a una cadena.

```
cadena = "Python"
print("Hola %s." % cadena)
```

`"Hola Python."`

Tuplas

Una tupla consiste en un conjunto de valores separados por comas. Son útiles para pares ordenados y retornar varios valores desde una función.

Creación

```
tupla = ()
otra_tupla = ("spam",)  # Nótese la coma
esta_tupla = 12, 89, 'a', True
esta_tupla = (12, 89, 'a', True)
```

Acceso

```
esta_tupla[0]
```
retorna `12`

Diccionarios

Un diccionario es un conjunto de pares `clave:valor` (o `nombre:valor`). Todos los nombres deben ser únicos.

Creación

```
diccionario = {}
diccionario = {'a':1, 'b':23, 'c':"huevos"}
```

Acceso

```
diccionario['a']
```
retorna 1

Borrado

```
del diccionario['b']
```

Búsqueda

```
diccionario.has_key("e")
```
retorna `False`

```
diccionario.keys()
```
retorna `['a', 'c']`

```
diccionario.items()
```
retorna `[('a', 1), ('c', 'huevos')]`

```
'c' in diccionario
```
retorna `True`

```
'otra_clave' in diccionario
```
retorna `False`

Listas: manipulación

Una de las estructuras de datos más importantes en Python son las listas. Además de ser muy flexibles, cuentan con varias funciones de control.

	Código	Valor de retorno	Contenido actual de la lista
Creación	`lista = [5, 3, 'p', 9, 'e']`		`[5,3,'p',9,'e']`
Acceso	`lista[0]`	`5`	`[5,3,'p',9,'e']`
Slicing	`lista[1:3]`	`[3, 'p']`	`[5,3,'p',9,'e']`
	`lista[2:]`	`['p', 9, 'e']`	`[5,3,'p',9,'e']`
	`lista[:2]`	`[5, 3]`	`[5,3,'p',9,'e']`
	`lista[2:-1]`	`['p', 9]`	`[5,3,'p',9,'e']`
Tamaño	`len(lista)`	`5`	`[5,3,'p',9,'e']`
Ordenar	`lista.sort()`		`[3,5,9,'e','p']`
Añadir elemento	`lista.append(37)`		`[3,5,9,'e','p',37]`
Retornar y remover	`lista.pop()`	`37`	`[3,5,9,'e','p']`
	`lista.pop(1)`	`5`	`[3,9,'e','p']`
Insertar	`lista.insert(2, 'z')`		`[3,'z',9,'e','p']`
Remover / eliminar	`lista.remove('e')`		`[3,'z',9,'p']`
	`del lista[0]`		`['z',9,'p']`
Concatenación	`lista + [0]`	`['z', 9, 'p', 0]`	`['z',9,'p']`
Búsqueda	`9 in lista`	`True`	`['z',9,'p']`

Listas: comprensión

Una expresión especial encerrada entre corchetes que retorna una nueva lista, de la siguiente forma:

```
[expresión for ítem in secuencia if condición]
```

La condición es opcional.

```
[x*5 for x in range(5)]                          [0, 5, 10, 15, 20]
[x for x in range(5) if x%2 == 0]                [0, 2, 4]
```

Definición de clases y funciones

Funciones

```
def mi_funcion(parametro1, parametro2):
    """Documentación de la funcion.
       Se accede desde mi_funcion.__doc__"""
    # Bloque de código indentado
    res = parametro1 + parametro2
    return res
```

Clases

```
class MiClase(ClaseDesdeLaCualHeredaOpcionalmente):
    def __init__(self):
        ClaseDesdeLaCualHeredaOpcionalmente.__init__(self)
        # Inicialización aquí
        self.algun_objeto = "hola mundo"
    def otra_funcion(self, argumento):
        if argumento == "alguna contradiccion":
            return False
        else:
            return True

mi_objeto = MiClase()
```

Archivos

Abrir:

```
archivo = open("carpeta/archivo.txt")  # Sólo lectura por defecto
```

Acceder:

```
archivo.read()
archivo.readline()
archivo.readlines()

for cada_linea in archivo:
```

Retorna el contenido del archivo.
Lee una línea del archivo.
Retorna una lista de cadenas (un elemento por cada línea).

Iterar entre las líneas del archivo.

Cerrar:

```
archivo.close()
```