THE PYTHON CHEATSHEET

ESTRUCTURAS

CADENAS DE TEXTO

s=’’ or s=””

Acción	Method	Comentarios
reemplazar	string.replace(s,’search’, ‘replace’)	
dividir	s.split(s,’sep’)	
encontrar	string.find(s,’search’)	Necesita ‘**import** string’ Retorna el índice de la primera ocurrencia
contar	string.count(s, ‘search’)	Número de ocurrencias
encontrar (regexp)	[m.start() for m in re.finditer('regexp', s)] [m.start() for m in re.finditer('(?=regexp)', s)]	requiere ‘**import** re’ para secuencias que se solapan
mayúsculas/minúsculas	s.upper()/s.lower()	Retorna la cadena en mayúsculas o minúsculas

LISTAS

a=[]

Acción	Método	Comentarios
acceso	a[i]	
rango	a[i:j]	
tamaño	len(a)	
eliminar	del a[i]	
añadir	f.append(v)	
ordenar	f.sort or sorted(f)	Más información aquí: https://wiki.python.org/moin/HowTo/Sorting
mezclar	‘glue’.join(a)	retorna ‘a[0]gluea[1]gluea[2]…’
Copia 'profunda'	a2=copy.deepcopy(f)	necesita ‘import copy’
pop	a.pop()	Retorna y elimina el último elemento de la lista
rango	range([s],e) range(e,s,-1)	retorna [s,s+1,s+2,…, e-1] retorna [s-1,s-2,…,e+1,e]
xrange	Como range	Retorna un iterador en su lugar (mejor para bucles con $>10^6$ iteraciones)
Valor único	list(set(a))	
diferencia	list(set(a)-set(b))	Retorna elementos en a que no están en b
índice	a.index(v)	Retorna la posición de la primera ocurrencia de v en a

DICCIONARIO

d={}

Acción	Método	Comentario
claves	d.keys()	
valores	d.values()	
acceso	d[k]	
asignación	d[k]=v	

COMENTARIOS

```
''' comentario en una única línea
# comentario en una única línea
''' comentario en
múltiples líneas '''
```

I/O

IMPRIMIR

```
print v     # puede ser un valor único o una estructura (e.g. string, list, dictionary)
```

FORMATO

```
'{0} cualquier texto {1} cualquier texto {2} …'.format(v0,v1,v2…)
      #retorna una cadena formada por los valores de las variables en vez de {n}
```

ARCHIVO

```
f=open(path, 'access')#access is usually 'r' or 'w'
```

Acción	**Método**	**Comentarios**
leer	f.readlines()	Retorna un array de strings
escribir	f.write(string)	Usa '\n' para la nueva línea
guardar	f.close()	

CONTROL

BUCLE

```
for index in list:
      do_lines                #la indentación marca qué está dentro del bucle

bucle en una línea: [do_line for index in list]      #los resultados se retornan en una nueva lista
      #esto es equivalente a un map flexible(ver http://www.bogotobogo.com/python/python_fncs_map_filter_reduce.php)

while(condition):
      do_lines
```

MÉTODO

```
def method(arguments):
      method_lines
      return value            #opcional

      yield: retorna en este punto, pero continuará en este punto en la siguiente llamada al método
              (ver http://www.prasannatech.net/2009/07/introduction-python-generators.html)
```

ESTADÍSTICAS

import numpy as np

Acción	**Método**	**Comentarios**
media	np.mean(a)	a is a list of numbers. nanmean to ignore NaNs
desviación típica	np.std(a)	nanstd para ignorar NaNs (Not a Number)
min/max	np.amin(a) / np.amax(a)	nanmin/nanmax para ignorar NaNs
percentil	np.percentile(a,g)	Calcula el q-ésimo percentil Más en: http://docs.scipy.org/doc/numpy/reference/routines.statistics.html
redondear a la baja/alta	np.floor(x)/np.ceil(x)	Valor entero por debajo o por encima
redondear	np.fix(a[,decimals])	Redondea un array al entero más cercano (o el número dado de decimales)
sum/prod	np.sum(a)/np.prod(a)	Suma/producto de todos los elementos en el arrayelements in the array Más en: http://docs.scipy.org/doc/numpy/reference/routines.math.html

NUMPY.ARRAY

import numpy as np

matrix	m=np.array([[1,2,3],[4,5,6]])	más en: http://docs.scipy.org/doc/numpy/reference/arrays.ndarray.html
dimensión	m.shape()	(2,3)
acceso	m[1,2]	elemento en la **segunda** fila y la **tercera** columna
	m[:,1]	toda la primera columna como un array
'porción'		
añadir	m=np.append(m,[34])	añade al final de la matriz
tabla	t=np.empty(#rows,dtype=[(“name”,”type”),…]	**dtype** es una lista con tantos pares como columnas. Cada par contiene el nombre de la columna y el tipo (a-caracter, f-real, i-entero) y el **tamaño** (en bytes) de los datos en él: np.empty([53,dtype=[("pos", "i4"),("text", "a10”)])
iniciar	t=np.zeroes(#rows,dtype=[(“name”,”type”),…]	Inicia cada elemento de la tabla con ceros
acceso	t[pos] for rows; t[“name”] for cols	
ordenar	t=np.sort(t,order=(”name”…))	Ordena t filas por la columna “name” (se pueden fijar varias columnas más)
buscar	np.where(t[“name”]==”pattern”)	
	np.where(m>5)	
	np.seachsorted(t[“name”], “pattern”)	Busca en una columna ordenada (más rápido que where)

numpy.array es un recubrimiento directo de un array de C, y su uso es recomendado con arrays largos (>10^6 elementos) para optimizar el uso de memoria

TIEMPO

```
import time
t0=time.clock()
operation_lines
print 'tardó  {0}s hacer la operación'.format(time.clock()-t0)
```

Autor original: Dr. Rodrigo Santamaría Vicente.
Fuente: http://vis.usal.es/rodrigo/documentos/bioinfo/avanzada/pythonCheatsheet.pdf