Cheatsheet Python 2.x

Estructura Básica

```
#!/usr/bin/python

# Importacion de modulos

import  sys
from os import path

“”” comentario de
varias
líneas
“””

# Funcion main
# Bloque python: indentado (sin begin ni end) y ‘:’ en
# línea anterior

 def main():
    print 'Hola mundo desde',sys.platform

# Si es el modulo principal, ejecutamos main

if __name__ == '__main__':
    main()
```

Tipos Simples

```
# Flotante

a = 3.45

# Entero

a = 500

# Cadenas

a = “Hello world”

# Booleanos

a = True

type(a) # Imprime <type ‘boolean’>
```

PyCharm

Consola Python: Tools > Run Python Console
Breakpoint: Clickar zona izquierda línea
Botón derecho en fuente: Ejecución y Depuración
F8 en depuración: Paso a paso
En ventana de debug: Debug>Variables botón derecho, podemos cambiarlas
Preferences > Python Interpreter: Ajuste del intérprete de python

Cadenas

```
myString = “Hola mundo”

myString[0] # ‘H’

myString[1:] # ‘ola mundo’

myString[-2] # ‘d’

myString[2:4] # ‘la’

len(myString) # longitud de la cadena

myString1 + myString 2 # concatenación

str(cualquierVariable) #conversion a cadena

myString.lower() # minusculas

myString.upper() # mayusculas

tokens = myString.split(“ “) # separacion

nuevaCadena = (“ “).join(tokens) # unión

myString = r’Hola mundo\n\n\n’ # Modo raw

# Formateo de cadenas

print “Esta cadena: %s , este entero %d , este flotante %f “
% (“cadena”, 5, 3.24)
```

Funciones

```
# Declaracion

def nombreFuncion(parametroA,parametroB,parametroC):
        print str(parametroA)
        print str(parametroB)
        print str(parametroC)
        returnString = parametroA + parametroB +
        parametro B)
        ……
        return(returnString)

# Llamada

print nombreFuncion(“a”,”b”,”c”) # “abc”
```

Control de Flujo

```
if condicion:
        bloque1
elif:
        bloque2
else:
        bloque3

# Comparadores

==,!=,>,>=,<,<=

# combinaciones booleanas

if a==”hola” and b<4.0 and not c==”adios”:
        bloque
```

Material preparado por Óscar Marín Miró (oscar@outliers.es)

python™

Cheatsheet Python 2.x

```
while(condicion):
        bloque

for i in range(100): # de 0 a 99
        bloque
for i in range(10,100): # de 10 a 99
        bloque
for i in range(10,100,10) :# de 10 a 99 en pasos de 10
        bloque
for i in range(100,10,-10): # de 100 a 11 en pasos de -10
        bloque

for elemento in lista: # Recorre los elementos de una lista
        bloque      # En orden
```

Listas

```
lista = [‘hola’,’adios’]
lista[0] # ‘hola’
len(lista) # 2

lista.append(“hasta luego”) # un elemento mas

lista.extend([‘hasta luego”,”nos vemos”]) # concateno listas

lista 1 + lista2 # otra manera de concatenar

listaNueva = [] # lista vacia, inserto con listaNueva.append
```

Tuplas

```
tupla = (1,2,”hola”)

len(tupla) # 3

tupla[2] # “hola”

tupla[1] = 3 # NO FUNCIONA, son inmutables

tupla.append(“otro”) # NO FUNCIONA

(x,y,z) = (1,2,3) # paso de lista a variables
```

Ordenación

```
a = [5,1,4,3]

sorted(a) # [1,3,4,5]  por defecto ascendente

sorted(a,reverse=True) # [5,4,3,1]

strs = [‘ccc’,’aaaa’,’d’,’bb’]

sorted (strs , key = len) # Con key ordeno por función

# [‘d’,’bb’,’ccc’,’aaaa’]

strs = ['xc', 'zb', 'yd' ,'wa']

# Devuelvo ultimo carácter

def MyFn(s):
        return s[-1]

sorted(strs, key=MyFn)  ## ['wa', 'zb', 'xc', 'yd']

alist = [4,3,5,10]

alist.sort()        # alist = [3,4,5,10]
```

Diccionarios

```
# Inicializo

dict = {}

dict[‘a’] = ‘alpha’
dict[‘g’] = ‘gamma’
dict[‘o’] = ‘omega’

dict[‘a’] # ‘alpha’

if ‘a’ in dict:
        print dict[‘a’] # ‘alpha’

# recorrido por claves

for key in dict: print key

# claves a lista

claves = dict.keys()
valores = dict.values()

# recorro claves y valores

for k,v in dict.items(): print k,v

del dict[‘a’] #borro clave

# diccionario: otra inicialización

dict = {'a':1, 'b':2, 'c':3}
```

Ficheros

```
f = open('foo.txt', 'r') # Fichero en modo read
# Recorro sus líneas e imprimo
for line in f:
        line = line.strip()
        print line
f.close()

f = open('foo.txt','a') # fichero en modo append: agrega

f.write("Una línea mas")
f.close()

# Fichero en modo escritura, crea de nuevo
fOut = open('fooNuevo.txt','w')
fOut.write(line)
fOut.close()
```

Módulos

```
# Modulo de calculo de factoriales. Nombre: 'factorial.py'

# Variables propias del modulo

numFunciones = 2

# Impresion del fact de n

def factorial(n):

    value = 1

    for i in range(1,n+1):
        value = value * i

    print value

# Retorno del fact de n

def factorial2(n):

    value = 1

    for i in range(1,n+1):
        value = value * i

    return value

def main():

    factorial(5)
    print factorial2(5)

# Cuando el modulo se ejecuta directamente, se ejecuta el
# main. Si se importa desde fuera, no se ejecuta

if __name__ == "__main__":
    main()

# IMPORTACION DEL MODULO ANTERIOR

# from factorial import factorial, factorial2
import sys
# Puedo indicar la ruta con la llamada siguiente
#sys.path.append('fact/')
import factorial

def main():
    print sys.path
    factorial.factorial(5)
    print factorial.factorial2(10)

    print factorial.numFunciones


if __name__ == '__main__':
    main()
```

Objetos

```
# Clase 'Point' de ejemplo en fichero 'point.py'

from math import sqrt

class Point:

    # El constructor. Todos los metodos necesitan 'self'
# como primer parámetro

    def __init__(self, x ,y):
        self.x = x
        self.y = y

    def modulo(self):
        return (sqrt(self.x*self.x+self.y*self.y))

    def moduloSquared(self):
        return(self.modulo() * self.modulo())

# Importacion de la clase en otro modulo

#!/usr/bin/python

import point

def main():

    myPoint = point.Point(3,4)

    modulo = myPoint.modulo()

    print myPoint.x
    print myPoint.y

    print "El modulo es %f" % modulo

    print "El modulo al cuadrado es %f" %
myPoint.moduloSquared()

if __name__ == '__main__':
    main()
```

Excepciones

```
#!/usr/bin/python


import sys


def main():

# Bloque try-except con captura de 'IOError'

    try:
        f = open("ficheroPrueba.txt")
    except IOError as error:
        print "No se pudo abrir el fichero... %s" % error
        exit(-1)

    for line in f:
        print line

    try:
        a = 70
        b = 0
        c = a/b
    except ZeroDivisionError as error:
        print "Se produjo el siguiente error: %s" % error
    #except:
    #    print "Unexpected error:", sys.exc_info()[0]
```

```
    else:
        print "El resultado es: %f" % c
    finally:
        print "Salgo..."
        exit(1)

if __name__ == '__main__':
    main()
```

Sistema

```
import os,time,sys,re,commands

def main():

    # Utilidades os

    # Listar directorios

    fileNames = os.listdir("/Users/oscarmarinmiro/Desktop")

    # Todos los ficheros

    for fileName in fileNames:
        print fileName

    # Solo los ficheros de texto

    for fileName in fileNames:
        match = re.search('.*\.txt$',fileName)
        if match:
            print match.group()

    command = "ps"

    (status,output) = commands.getstatusoutput(command)

    print output

    # Utilidades time

    # Dormir durante 10 segundos y tomo tiempos antes y despues

    tiempo1 = time.time()

    print "Tiempo1: %d" % tiempo1

    time.sleep(10)

    tiempo2 = time.time()

    print "Tiempo2: %d" % tiempo2

    timedelta = tiempo2 - tiempo1

    print "Tiempo transcurrido: %d" % timedelta


    return


if __name__ == '__main__':
    main()
```

Expresiones Regulares

```
#!/usr/bin/python

import re

def main():

    # La cadena

    cadena = "El perro se comio al gato y despues otros gatos se comieron al perro"

    # ojo, hace match de gato y gatos
    results = re.findall(r'gato',cadena)

    # Hace match de gato pero no de gatos
    # \b es 'frontera de palabra'

    results = re.findall(r'\bgato\b',cadena)


    cadena = "El perro se comio al gato y despues otros gatos se comieron al perro al grito de arigato"

    # '?' es un cuantificador que significa 0 o 1 apariciones

    results = re.findall(r'\bgatos?\b',cadena)

    for result in results:
        print result


    # Queremos encontrar telefonos en formato "NN-NNNNNNN

    # \d hace match con números
    # {2} quiere decir que hace match si hay 2 apariciones

    results = re.findall(r'\d{2}-\d{7}',cadena)


    print "Direcciones de correo:"

    # Queremos encontrar direcciones de correo
    # \w hace match con alfanuméricos

    results = re.findall(r'\b\w+@\w+\b',cadena)
```