PYTHON CHEAT SHEET

Rapido documentazione di consultazione per istruzioni, funzioni e tipi in Python

OUTPUT

```
print “Ciao”                                        #stampa una stringa
print x,y                                    #stampa le variabili x e y
print “%s . . . %s” %(x,y) #stampa in maniera formattata le variabili x,y
```

VARIABILI

```
#Una stessa variabile può farsi assegnare diversi tipi differenti
a = ‘Parrot Sketch’                                 #assegnazione stringa
a = 3                                               #assegnazione intero
a = 1.23                                            #assegnazione float
```

STRINGHE

```
s = r’c:\nome\ciao’   #Una stringa raw viene valutata senza tener conto
                      #dei caratteri di escape o altri caratteri speciali
s1 + s2          #concatenazione di stringhe es: ‘ci’ + ‘ao’ = ‘ciao’
s * 3            #ripetizione di stringa es: ‘ciao’ * 3 = ‘ciaociaociao’
len(s)                                              #lunghezza della stringa
s[i]                       #il carattere in posizione i della stringa s
s[i:j:k]                   #sottostringa da indici i a j ogni k-passi
p = ‘ababab’
p[0]             # Risultato: ‘a’
p[0:4]           # Risultato: ‘abab’
p[0:5:1]         # Risultato: ‘bb’
p.find(‘a’)                                 #cerca ‘a’ nella stringa p
p.replace(‘: (’, ‘: D’)    #sostituisce tutti i ‘: (‘ contenuti nella
                           #stringa p con ‘: D’
p.split(‘,’)     #restituisce una lista contenente tutte le sottostringhe
                 #comprese che iniziano e/o finiscono con ‘,’
```

```
S = ‘1,2,3’
S.split(‘,’)    # Risultato: [‘1’, ‘2’, ‘3’]
p.isDigit()          #ritorna True se la stringa è composta solo da
                     #cifre numeriche
p.lower()                      #restituisce la stringa p in minuscolo
p.upper()                      #restituisce la stringa p in maiuscolo
s = u’spam’                    #stringa codificata in Unicode
int(‘42’)                      #trasforma la stringa ’42’ in un intero
float(‘2.3’)              #trasforma la stringa ‘2.3’ in un decimale
ord(‘s’)                       #ritorna il valore ASCII del carattere
chr(64)         #ritorna il carattere ASCII corrispondente all’intero
```

COSTRUTTI DI CONTROLLO

```
if test:
     #...codice
elif other_test:
     #...codice
else:
     #...codice

while test:
     #...codice

#Costrutti per la manipolazione dei costrutti
break #esce dal blocco superiore
pass #non fa nulla

#ciclo while che si comporta come un ciclo while in C
x = next()                     # while((x = next())!=NULL)
while x:                       # {
     #...                       # //...
     x = next()                # }
```

```
V = [1, 2, 3]
for x in V:                    #per ogni x in V
     print x

range(5)                       #genera un vettore da 0 a 4
               #Risultato: [0, 1, 2, 3, 4]
range(2, 5)                    #genera una vettore da 2 a 4
               #Risultato: [2, 3, 4]
range(0,10,2)                  #genera un vettore da 0 a 9 ogni 2
               #Risultato: [0, 2, 4, 6, 8]
e = enumerate(v)     #Restituisce un oggetto iterativo che ad ogni passo
                     #ritorna una tupla (indice, v[indice]) attraverso
                     #il metodo next().
Z = zip(v1, v2)      #Accoppia ogni valore di v1 con il valore alla
                     #posizione di v2
#Esempio:
v1 = ['a', 'b', 'c']
v2 = [1, 2, 3]
e = enumerate(v)
e.next() #(0, 'a')
e.next() #(1, 'b')
e.next() #(2, 'c')
z = zip(v2, v1) #z = [(1, 'a'), (2, 'b'), (3, 'c')]
```

LISTE

```
L1 = [1, 2, 3]                              #assegnazione lista
L2 = L1        #copio il riferimento di L1 in L2, non il suo contenuto
L1[0] = 24               #assegno nella posizione 0 di L1 il valore 24
L2 = L1[:]                          #copio il contenuto di L1 in L2
len(L)                                      #lunghezza della lista
L + [1, 3]               #ritorna la concatenazione di L con [1,3]
L.append('different')                       #aggiunge in coda a L
```

```
L.pop(i)              #ritorna e rimuove l'elemento di indice in in L
L.sort()              #ordina la lista
L.reverse()           #inverte la lista
M = [ [0, 1, 2],
      [3, 4, 5],
      [6, 7, 8] ]
M[0]                                        #ritorna la riga 0 di M
M[1][2]               #ritorna l'elemento di riga 1 e colonna 2 di M
L1 == L2                        #confronta il contenuto di L1 con L2
L1 is L2                        #confronta il riferimento di L1 con L2
del L[0]                             #elimina l'elemento di indice 0
del L[0:5]      #elimina tutti gli elementi compresi tra gli indici 0 e 4
```

DIZIONARI

```
D1 = { 'K1':1, 'K2':2 }
D1['K1']        #restituisce il valore corrispondente alla chiave 'K1'
D1.has_key('K1')           #torna True se D1 contiene una chiave 'K1'
'K1' in D1                                              #come sopra
D1.keys()                           #ritorna una lista di chiavi di K1
D1.values()                         #ritorna una lista di valori di K1
D1.copy()                                    #crea una copia di D1
elem = D1.get(key, default)     #restituisce il valore corrisponde a key
                                #se il valore non c'è, torna default
len(D1)                                  #lunghezza del dizionario
D1['question'] = 42   #Sostituisce il valore corrisponde a 'question' con
                      #42. Se 'question' non è una chiave di D1, viene
                      #creata al momento.
D1.items()            #Restituisce una tupla di coppie key, value
```

FILE

```
fp = open('file.dat', 'w')      #Apre il file in modalità scrittura
#Modalità:
#     w -> scrittura            w+ -> scrittura e lettura
#     r -> lettura              r+ -> lettura e scrittura
#     a -> append               a+ -> append e lettura
#     w, a, w+, a+ -> se il file non esiste viene creato
#     w, w+ -> se il file esiste, viene svuotato
#     r, r+ -> se il file non esiste, open torna None
#     a, a+ -> se il file non è vuoto, il contenuto nuovo finisce
#               in coda a quello vecchio

fp.readline()                              #legge una riga del file
fp.readlines()  #torna una lista i cui elementi sono le righe del file
fp.write(str)                         #scrive la stringa str nel file
fp.writelines(lst)   #scrive ogni elemento della lista come riga nel file
fp.close()                                          #chiude il file
```

FUNZIONI

```
def function(arg1, arg2):
     #...codice
     return res
```