Programmazione in Python per la bioinformatica

University of Verona

Sommario

- Scrivere ed utilizzare i moduli
- Scrivere ed utilizzare i package

Cosa sono i moduli

Cosa e' un modulo e a cosa serve

- *Modulo:* file python con estensione *.py* che contengono definizioni di variabili o funzioni.
- Tipicamente i moduli racchiudono elementi che hanno a che fare con un determinato argomento.
- Raggruppare il codice in moduli aiuta a rendere il codice più semplice da leggere e ri-utilizare.

Esempio di modulo

dnautil.py

```
"""
modulo che contiene alcune funzioni di utilita' per le sequenze di dna
"""

def has_stop_codon(dna,frame=0):
    """
    Controlla se una stringa di DNA contiene
    un codone di terminazione
    (i.e., una tripletta 'tga','tag','taa')
    """
    stop_codons = ['tga','tag','taa']
    for i in range(frame,len(dna),3):
        codon = dna[i:i+3].lower()
        print(codon)
        if codon in stop_codons:
            return True
    else:
        return False


def revcompl(dna,rules={'a':'t','c':'g','g':'c','t':'a'}):
    """
    dna rappresenta la stringa di dna e rules un dizionario che associa nucleotidi a nucleotidi
    la funzione deve ritornare la stringa complementare inversa
    """
```

Usare i moduli

Come richiamare un modulo

- *import dnautil* carica il modulo per poterlo utilizzare
- ATTENZIONE il file deve trovarsi nella stessa directory
- ATTENZIONE non posso accedere direttamente alle funzioni definite ma devo usare *dnautil.has_stop_codon()*

```
>>> import dnautil
>>> my_dna = 'atgtaa'
>>> dnautil.has_stop_codon(my_dna)
atg
taa
True
>>> has_stop_codon(my_dna)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'has_stop_codon' is not defined
>>>
```

Dove si trovano i moduli

Come python richiama un modulo

Quando un modulo viene importato python cerca per prima cosa un modulo *buil-in*. Se il modulo non viene trovato python cerca un file .py in:

1. la directory corrente
2. la directory dove python e' stato installato
3. le directory indicate dalla variabile di ambiente PYTHONPATH

```
>>> import sys
>>> sys.path
['', '/home/alessandro/catkin_ws/devel/lib/python2.7/dist-packages',
 '/opt/ros/groovy/lib/python2.7/dist-packages', '/home/alessandro/do
c/Didattica/Verona/svnDidattica/python/material/pythonForGenomicData
Science--coursera/code/test', '/home/alessandro/doc/Didattica/Verona
/svnDidattica/python/material/pythonForGenomicDataScience--coursera/
code/quiz', '/usr/lib/python3.2', '/usr/lib/python3.2/plat-linux2',
'/usr/lib/python3.2/lib-dynload', '/usr/local/lib/python3.2/dist-pac
kages', '/usr/lib/python3/dist-packages']
>>>
```

Estendere il path

Possiamo includere nuove dirctory nel path

sys.path.append('NUOVA-DIR')

Per rendere il cambiamento permanente dobbiamo cambiare il PYTHONPATH (e.g., editando .bash_profile)

```
>>> sys.path.append('nuova-dir')
>>> sys.path
['', '/home/alessandro/catkin_ws/devel/lib/python2.7/dist-packages',
 '/opt/ros/groovy/lib/python2.7/dist-packages', '/home/alessandro/do
c/Didattica/Verona/svnDidattica/python/material/pythonForGenomicData
Science--coursera/code/test', '/home/alessandro/doc/Didattica/Verona
/svnDidattica/python/material/pythonForGenomicDataScience--coursera/
code/quiz', '/usr/lib/python3.2', '/usr/lib/python3.2/plat-linux2',
'/usr/lib/python3.2/lib-dynload', '/usr/local/lib/python3.2/dist-pac
kages', '/usr/lib/python3/dist-packages', 'nuova-dir']
```

Importare i nomi di un modulo

Come importare tutti i nomi di funzioni e variabili in un modulo

- from NOME-MODULO import *

```
>>> from dnautil import *
>>> has_stop_codon('acttagtaa')
act
tag
True
```

Package

Cosa sono i package

- Package raggruppano moduli
- Un package in python e' una directory che contiene i file relativi ai moduli del package
- la directory deve contenere un file speciale *__init__.py* che indica che la directory e' un paclage python.
- il file __init__.py puo' essere vuoto oppure puo' contenere codice di inizializzazione per il package.

```
bioseq
├── dnautil.py
├── __init__.py
├── proteinutil.py
└── __pycache__
```

Package annidati

I package possono essere annidati

- Posso avere package all'interno di altri package

```
bioseq
├── dnautil.py
├── fastautil
│   ├── fastautil.py
│   └── __init__.py
├── __init__.py
├── proteinutil.py
└── __pycache__
    ├── dnautil.cpython-32.pyc
    ├── __init__.cpython-32.pyc
    └── proteinutil.cpython-32.pyc
```

Importare package

Possiamo importare moduli e funzioni dai package

import bioseq.dnautil

from bioseq import dnautil

from bioseq.dnautil import has_stop_codon

```
>>> import bioseq.dnautil
>>> bioseq.dnautil.has_stop_codon('atc')
atc
False
>>> from bioseq import dnautil
>>> dnautil.has_stop_codon('atc')
atc
False
>>> from bioseq.dnautil import has_stop_codon
>>> has_stop_codon('atc')
atc
False
```

Esercizi su moduli e packages

Eserc mod Q1

Q1 Scrivere un modulo eserc_mod_Q1.py che contiene due funzioni: i) generate_rnd_dna(n,alphabet='atgc') che crea una string di n caratteri scegliendo in maniera randomica i caratteri dall'alfabeto passato come parametro; ii) analyse(dna,alphabet='acgt') che ritorna un dizionario che rappresenta la composizione della stringa (i.e., numero di occorrenze per ciascun carattere dell'alfabeto) Suggerimento: usare il modulo *random* ed il metodo *choice*. Importare il modulo dall'interprete e poi eseguire la funzione test(). Scaricare e modificare eserc_mod_Q1.py [Sol: eserc_mod_Q1.sol]

Esercizi su moduli e packages

Eserc mod Q2

Q2 Scrivere una modulo eserc_mod_Q2.py che contenga tre versioni della funzione count(dna,base). La funzione conta quante volte la base si trova nella string dna. La prima versione count_for(dna,base) usa un ciclo for per scandire tutti i caratteri di dna e contare le occorrenze di base. La seconda count_while(dna,base) esegue un while. La terza count_direct(dna,base) usa la funzione count delle stringhe. Dall'interprete importare il modulo ed usare la funzione test_run_time() per vedere quale e' piu' veloce. Scaricare e modificare il file eserc_mod_Q2.py [Sol: eserc_mod_Q2.sol]

Esercizi su moduli e packages

Eserc mod Q3

Q3 Creare un package bioseq che contiene due moduli: danutil e proteinutil. dnautil contiene le funzioni has_stop_codon e la funzione revcompl. proteinutil contiene la funzione per decidere se una proteina e' valida ed una funzione per correggere una proteina che contiene caratteri non validi. Scrivere uno script test_bioseq.py che chiede all'utente se desidera inserire una proteina o una stringa di dna. Nel primo caso lo script chiede la stringa e stampa se ha un codone di stop ed il complemento inverso (usare frame 0 e le regole standard di sostituzione di nucleotidi per i valori di default). Se la stringa e' una proteina, lo script chiede la stringa della proteina ed una stringa di caratteri validi. Lo script stampa quindi la proteina corretta [Sol: bioseq.zip, test_bioseq.sol]