Micro-dispense

Digital Humanities Winter School Palermo 2019
Laboratorio di programmazione “Analisi del testo con Python”
Università di Palermo, 4-7 marzo 2019
Prof. Paolo Monella

Indice

Cos’è questo file?

Durante il laboratorio, userò questo file come una lavagna digitale per condividere con voi link, frammenti di codice ed altro testo. Contiene anche gli esercizi e link ad altri materiali.

Alcuni indirizzi web utili

- Winter school: https://dhwspa19.unipa.it/
- Laboratorio programmazione
  - Syllabus: https://dhwspa19.unipa.it/syllabus-laboratorio-programmazione/

    - Cosa preprare per i laboratori: https://dhwspa19.unipa.it/cosa-preparare-per-i-laboratori/ (sezione *Laboratorio programmazione*)
    - Cartella condivisa Google Drive coi materiali del laboratorio: http://tinyurl.com/labprogrammazione2019
        - Contiene il file con queste micro-dispense

Tenendo aperto questo file, puoi avere tutto questo, pronto da copiare/incollare, in tempo reale direttamente sul tuo computer.

Anna ed Elsa

- Vai su Code.org, tutorial *Programma con Anna ed Elsa*: https://studio.code.org/s/frozen/stage/1/puzzle/1.
    - Ricorda che “gira a destra di 90 gradi” vuol dire “vòltati verso destra (stando però ferma, senza camminare) di 90 gradi”
    - Alla fine di ogni esercizio, clicca su “Mostra il codice” prima di passare al successivo
- Punti-chiave:
    - ***Esercizio 4***: cicli, e argomenti delle funzioni (quante volte ripetere il ciclo)
    - *Esercizio 5*: cicli di cicli (3 quadrati)
    - *Esercizio 7*: argomenti delle funzioni (avanti e indietro)
    - *Esercizio 13*: disegna un cerchio (360 gradi)
    - ***Esercizio 14***: funzioni (cerchio)
    - *Esercizio 16*: argomenti delle funzioni (cerchio di dimensione x)

Esercizi (o: La via per la gloria)

Disclaimer: alcuni dei seguenti esercizi potranno produrre risultati non adatti ad un pubblico di studiosi seri. Gli esercizi in grigio sono opzionali.

Esercizi sulle variabili (V)

V1. Crea una variabile “nome” con il nome del tuo animale domestico, o il tuo soprannome, o un nome buffo; crea un’altra variabile “eta” (senza accento) con la tua età (sei libero di sottrarre quanti anni vuoi all’età reale, se hai la mia età); crea una terza variabile “cognome” con il tuo cognome.

V2. Stampa a schermo le tre variabili, una alla volta.

V3. Concatena (sopran)nome e cognome in modo da ottnere qualcosa come “Pontrilla Monella” o “Fuffy Monella” (occhio allo spazio tra i due!). Abbiamo creato un nuovo, mirabolante, membro della tua famiglia. Stampa il risultato a schermo (d’ora in poi, semplicemente “stampa”).

V4. Stampa l’iniziale del soprannome (nel caso di “Fuffy”), stampa “F”.

V5. Stampa in una nuova riga solo le iniziali del tuo nuovo parente (ad esempio, nel caso di “Fuffy Monella”, stampa “FM”. Se ti senti in vena, stampa qualcosa come “F. M.” (occhio al punto più spazio in mezzo, e al punto finale).

V6. Stampa “Che nome breve! È lungo solo [numero di lettere del (sopran)nome] lettere!”. Ricorda che la funzione built-in `len()` conta (anche) la lunghezza in caratteri di una stringa. Occhio però: `len()` restituisce un intero; ci vuole `str(len(qualcosa))` per trasformare l’intero in una stringa e poterla concatenare ad altre stringhe.

V8. Crea una nuova variabile “futuro”, e dalle come valore il doppio dell’età. L’operatore per la moltiplicazione è l’asterisco *.

V9. Mettiamo che la mia età sia 41 anni (non ho tolto niente!) e il mio nuovo parente sia Fuffy Monella. Stampa (concatenando tutto): “All’età di [il doppio della tua età] anni [nome del tuo nuovo parente] [cognome del tuo nuovo parente] troverà il senso della vita”. Io, ad esempio, stamperò: “All’età di 82 anni Fuffy Monella troverà il senso della vita” (al vostro parente andrà sempre meglio che a me). Attenzione: la variabile “futuro” contiene un (numero) intero. Per trasformare il numero in una stringa, e poterla concatenare col resto della frase, ricorda che `str(5)` restituisce una stringa “5” (non è più un “intero”, ma una “stringa”), e che `str(futuro)` restituirà lo stesso numero, ma gestito come una stringa.

Esercizi sulle liste (L)

L1. Crea una lista “L” (maiuscolo) che contenga 3 o 4 ingredienti di una pozione magica che fa diventare intelligenti gli stupidi e stupidi gli intelligenti. Metti in cima alla lista l’ingrediente segreto, il più importante. Stampa la lista.

L2. Stampa l’ingrediente segreto, il primo della lista (ricordi gli indici di una lista? Occhio, iniziano da 0, non da 1!).

L3. Stampa “[nome del tuo nuovo parente], quando torni porta Nutella e [primo ingrediente della lista]. Nel mio caso: “Fuffy, quando torni porta Nutella e olio usato della friggitoria delle arancine”. Fantastico. Sii fiero di te. Ed è solo l’inizio.

L4. Salva in una variabile “numin” il num(ero) di in(gredienti) della pozione (cioè della lista), e stampa: “[nome del tuo nuovo parente], hai comprato tutti e [numero ingredienti] gli ingredienti?”. Ad esempio, qualcosa come: “Fuffy, hai comprato tutti e 4 gli ingredienti?”.

Esercizi sui cicli con `for` (C)

C1. Stampa ogni ingrediente della pozione in una riga. Cioè: *per ogni* (`for`) ingrediente in (`in`) lista, stampalo. Ricorda i due punti alla fine della riga del `for`, e l’indentazione nel blocco della/e istruzione/i successivea/e.

C2. Quasi lo stesso: *per ogni* (`for`) ingrediente, stampa “[Nome del tuo nuovo parente], ti sei ricordato di comprare [ingrediente]? -- No -- Come no?”.

C3. Aggiungi un ingrediente alla lista. Attenzione! Viene dalla magia nera. Ha la virtù di trasformare una pozione banale in una radical chic. Si tratta dello zenzero. Qui torna utile la funzione `nomelista.append(elementonuovo)`. Ricorda che `elementonuovo` può anche essere una semplice stringa.

C4. Rifai ora l’esercizio C2. Cioè: copia e incolla in fondo allo script il ciclo che avevi creato in quell’esercizio. La lista include ora anche l’ingrediente del signore oscuro?

Esercizi sulla tokenizzazione (divisione in parole) con `split()` (T)

T1. Visualizza nel tuo browser il documento TEI XML che stai preparando nel laboratorio di markup (d’ora in poi “sorgente XML”), facendoci doppio clic sopra. Copia quel che si visualizza nel browser. Crea una variabile “t” (“t” come “testo”) e dalle come valore il testo che hai copiato dal browser (o una parte di esso, se pensi che sia davvero troppo lungo; fino a una cinquantina di righe, però, direi di tenerlo tutto). Stampa il tuo testo.

T2. Piccolo trucco, grande risultato: nella prima riga dello script (dopo eventuali commenti), incolla questo codice:

```
from string import punctuation
```

Poi, alla fine dello script, nell’ultima riga, copia/incolla queste tre righe di codice (attenta all’indentazione della seconda riga; la terza riga *non* è indentata):

```
for punt in punctuation:
    t = t.replace(punt, '')
t = t.lower()
```

La terza riga rende lowercase, cioè minuscolo, l’intero testo. Questo ci potrà tornare utile quando creeremo il dizionario macchina. Se non vuoi che il testo sia tutto minuscolo, non incollare la terza riga `t = t.lower()`

T3. Tokenizza il testo contenuto nella variabile “t”, cioè tokenizza il testo. In pratica, crea una nuova lista “P” (maiuscolo) con tutte le parole del testo, un elemento per parola. Ricorda la funzione `stringa.split()`. Stampa la lista così com’è (senza usare un ciclo).

T4. Miglioriamo ora il codice dell'esercizio T2. Dopo aver creato la lista P con le parole del testo, stampa ogni parola in una riga (usa un ciclo con `for`). Per ogni [parola] in P, stampa la parola.

T5. Aggiungi una riga al tuo script per fargli stampare: "Il testo è di [numero di parole] parole". Ricorda che la funzione `len(lista)`, in questo caso `len(L)`, conta quanti elementi ci sono in una lista.

T6. Crea un dizionario macchina (cioè un indice delle parole) del tuo testo. In pratica: crea una variabile "S" (maiuscolo) e dalle come valore le parole del tuo testo, ma prese una volta sola. Ricorda che questo è quel che fa la funzione `set(nomelista)`. Stampa l'indice (cioè S).

T7. L'indice S non è in ordine alfabetico, eh? Magia: la fuzione `sorted()` ordina il suo argomento, cioè mette in ordine alfabetico cioè che contiene tra parentesi. Prova a creare una nuova variabile O (maiuscolo) che contenga sorted(S), stampala e vedrai il sogno diventare realtà.

T8. Alla fine, stampa una riga "Il vocabolario del testo è composto da [lunghezza dell'indice, o dizionario macchina] termini, mentre il testo è lungo [lunghezza della lista di parole] parole."

T9. Ora si fa sul serio! Calcoliamo la varietà lessicale, misurata (un po' grossolanamente, ma tant'è) come il rapporto tra il numero di termini univoci nell'indice e il numero totale di parole. Stampa "L'indice di varietà lessicale è [risultato della divisione]". In pratica: crea una variabile "v" (minuscolo) che contenga il risultato di tale divisione, e poi stampala. Ricorda che l'operatore di divisione in Python è la barra posta sopra il 7 (/).

T10. Yahoo! Analisi stilometrica! Analisi stilometrica!

Esercizi sull'analisi del sorgente TEI XML con `lxml` (X)

X1. Crea un nuovo script, e chiamalo "myparsing.py". In cima (dopo eventuali commenti iniziali), copia/incolla queste righe di codice:

```
from lxml import etree
ns = {'t': 'http://www.tei-c.org/ns/1.0',            # per
TEI XML
'xml': 'http://www.w3.org/XML/1998/namespace',  # per
attributi come xml:id
'h': 'http://www.w3.org/1999/xhtml'}            # per l'HTML
```

X2. Importa e parsa il tuo file TEI XML. Chiama "tree" l'albero XML parsato.

X3. All'interno del tuo albero, trova gli elementi con nome "body" e conservali in una variabile "B" (maiuscolo). Poi stampa cioè che hai che ha trovato (come semplice lista). Ricorda che, per la questione dei namespace, il comando `findall()` deve avere questo aspetto:
B = `tree.findall('.//t:body', ns)`

X4. Altra cosa da tenere a mente: `findall()` restituisce sempre una lista. In questo caso, c'è un solo elemento `<body>` nel file, ma in futuro troveremo più elementi. Quindi migliora lo script così: invece di stampare direttamente la lista, crea un ciclo con `for` che stampi ogni elemento in una riga a sé (vedi l'esercizio C1).

X5. Cambia lo script in modo da trovare un tipo di elemento (per nome elemento) di tua scelta e stampa i risultati, ognuno in una riga (con `for`). Insomma, lo stesso dell'esercizio, ma con un elemento a tua scelta.

X6. Il "metodo" `.tag`, attaccato alla fine di una variabile che contiene un elemento, restistuisce il tag (il nome) di quell'elemento. Ora, invece di stampare riga per riga l'"elemento" in astratto, stampa il suo nome.

X7. Cerca elementi con un certo nome che abbiano un certo attributo (al di là del valore di quell'attributo). Ad esempio, elementi `<distinct>` che abbiano un attributo `@type`. Stampa, riga per riga, i nomi degli elementi trovati (il risultato non dovrebbe essere una gran sorpresa). Ricorda la sintassi
B = `tree.findall('.//t:distinct[@type]', ns)`

X8. Lo stesso dell'esercizio precedente, ma non stampare il nome di ogni elemento, ma il valore dell'attributo ricercato. Nell'esempio sopra, i valori di `@type`. Ricorda il "metodo" `.get("nomeattributo")` "attaccato" alla fine di una variabile che contenga un elemento.

X9. Cerca elementi con un certo nome che abbiano un certo attributo con un certo valore. Ad esempio, elementi `<distinct>` che abbiano un attributo `@type` con valore `"etym"`:
`B = tree.findall('.//t:distinct[@type="etym"]', ns)`
Ricorda che le virgolette interne (qui `"etym"`) devono essere doppie, per non confondersi con le esterne, che sono singole.

X10. Lo stesso di X6 (trova elementi per nome del tag), ma stavolta stampa, riga per riga, non il nome dell'elemento (beh, lo sapevamo già…) ma il suo contenuto testuale. Ricorda il "metodo" `.text`

X11. Cerca ora (e salva nella lista B) gli elementi <l> (i versi). Falli contare a len() e salva il risultato in una variabile "nv" (minuscolo: numero versi). Stampa: "Nel testo ci sono [numero di versi] versi".

X12. Di séguito, per ogni verso (per ogni "line" in B), salva (temporaneamente) il suo contenuto testuale in una variabile "tv" (testo del verso, non il rettangolo del signore oscuro) usando il "metodo" `.text`

X13. Saliamo di livello! Per ogni verso in B:
dividi in parole il suo contenuto testuale (cioè "tv") con `.split()` e salva la lista risultante in "P" (maiuscolo, per *parole*). È lo stesso che hai fatto nell'esercizio T3; stampa la lista P di quel verso.

X14. Alla grande! Miglioriamo ancora lo script! Invece di stampare la lista P di ogni verso, salva la sua lunghezza `len(P)` in una variabile "np" (numero parole) stampa quella (...cioè in pratica la sua lunghezza in parole!) Ci siamo quasi...

X15. ...e il gioco è fatto! Ora abbiamo il numero di versi "nv" e, per ogni verso, il suo numero di parole "np". Ora ci serve un modo per sommare tutti i numeri di parole insieme. Subito dopo le righe

```
from lxml import etree
ns = {'t': 'http://www.tei-c.org/ns/1.0',            # per
TEI XML
'xml': 'http://www.w3.org/XML/1998/namespace',  # per
```

```
attributi come xml:id
'h': 'http://www.w3.org/1999/xhtml'}            # per l'HTML
```
crea (inizializza) una variabile “cp” (conteggio parole), e dalle valore iniziale 0 (zero). È sempre un inizio...

X16. Adesso, dentro il ciclo
```
for verso in B:
    print(verso.text)     # Esercizio X12
    P = verso.split()     # Esercizio X13
    np = len(P)           # Esercizio X14
    print(np)
```
dopo l’ultimo comando `print(np)`, chiediamo a Python di aumentare il conteggio delle parole “cp”, sommandogli il numero di parole di quel verso “np”. Ad esempio, se il primo verso ha 6 parole, e il secondo 7, all’inizio del ciclo “cp” sarà 0; alla prima ripetizione del ciclo “cp” sarà 0 + 6, quindi 6; alla seconda ripetizione, “cp” sarà 6 + 7, quindi 13 -- se non sbaglio -- (l’operatore per la somma, in Python… è “+”, chi se lo sarebbe aspettato!): qualcosa come
```
    cp = cp + …           # Completa tu, sostituendo i puntini
```

X17. La vittoria è vicina! Alla fine di tutte le ripetizioni dello script, “cp” conterrà il numero totale di parole di tutti i *versi* (giustamente escludendo titoli `<head>`, commenti, numeri di pagina, etc.). Calcola ora la media di parole per verso (numero totale di parole *diviso* numero di versi) *al di fuori del ciclo* (togli l’indentazione) e conservala in una variabile “media”. Stampa ora “La media di parole per verso è [media]”. Ricorda che l’operatore per la divisione è “/”, e che il risultato sarà un numero, non una stringa, quindi torna utile l’amico `str()`, quindi qualcosa come:
```
print( "La media di parole per verso è " + str(media) )
```
Victory is yours! Hai fatto una mini-analisi stilometrica con Python! Lode e onore a te.

X18. Per accontentare i superstiziosi e non chiudere con 17 esercizi, e per marcare la maggiore età della tua competenza pythonica, ecco un gran finale (opzionale): cerca tutti gli elementi `<rs>` (nomi) del file romualdo.xml, conserva il risultato in una lista N, e per ogni elemento trovato (ciclo con `for`) stampa il suo contenuto testuale (il nome della persona).

X19. Sei arrivato fin qui? Beh, a questo punto tanto vale che trasformi la lista N in un indice (dizionario macchina) dei nomi con `set(N)` , ordini alfabeticamente l’indice con `sorted(set(N))` e chiedi al ciclo (`for`) di stampare ogni nome in ogni riga:
```
for nome in sorted(set(N)):
    print(nome)
```
...ed entrerai così per sempre nel Walhalla degli eroi che hanno creato un vero indice dei nomi con Python! (Sento dire che regalano un’arancina ad ogni nuovo eroe).

Ciò che ho scritto sullo schermo

Martedì 5 marzo 2019

C++
Inteprete Python
traduce codice Python in codice C++
Funzioni built-in dell'interprete Python

print()

modulo = insieme di funzioni per uno scopo

lxml

.findall()

editor

Anaconda
1) Inteprete Python
2) Alcuni moduli (tra cui lxml)
3) Alcuni editor Python tra cui Spyder

angolo(100)

--⟶
1. Vai dritto (x)
2. Gira a destra (90°)
3. Vai dritto (x)

100 è l'argomento della funzione

Esercizio 1: Scrivi il contenuto della variabile sullo schermo

print()
a

stringa = una sequenza di caratteri

0. Scrivi:
a = '5'
1. Definisci una variabile b che contenga il doppio di a
2. Stampa la variabile b

Raddoppia a e stampala

Python: Lista
C++: array
JS: array

Initialize a list

Crea una variabile ‘l’ che contenga la lunghezza della lista arancina.
Stampa la variabile ‘l’

Salva in una variabile “l” il num(ero) di in(gredienti) della ricetta; poi stampa: “[nome del tuo nuovo parente], hai comprato tutti e [numero ingredienti] gli ingredienti?”. Ad esempio, qualcosa come: “Fuffy, hai comprato tutti e 4 gli ingredienti?”.

Mercoledì 6 marzo 2019

Prova uno due
Si legge bene in fondo?
Userò questa come lavagna digitale

Stampa a schermo:
“L’indice di varietà lessicale del testo è 0,87”
Dove tale indice è la divisione tra il numero di parole uniche (type) e il numero totale di parole.
Il “diviso” si fa con /

Se do la funzione ‘findall’ gli elementi <l> nel tree, il risultato sarà
verde una stringa
giallo un (numero) intero
rosso una lista

Funzionamento di findall:

Crea una lista L che contenga l'output (il risultato) di
NOMEALBERO.findall()

Giovedì 7 marzo 2019

dhwspa19.unipa.it
---→ menu 'laboratori'
→ Lab. programmazione
----→ sezioni 'Materiali'
→ Cartella Google Drive
→ boncompagno.xml
→ romualdo.xml

inglese:case
italiano:case

Fagli cercare tutti gli elementi <l> con namespace TEI (t).
Salva la lista in una variabile L (maiuscolo)

Per ogni verso <l> nella lista L dei risultati, stampalo.

Behnel
<l>Ciao a tutti</l>

Per ogni verso <l>, crea una variabile (temporanea) "n" che contiene il numero di parole di quel verso, **e stampa "n"**.
Ricorda che:
nomestringa.split()
crea una lista con un
elemento per parola
len(L)
calcola la lunghezza di una
lista

Esercizio di ammissione al Wahlalla dei gloriosi programmatori:

Fagli scrivere
"*La media di parole per verso è 4.375"*

Poi noi lo mandiamo a Odino e ti arriverà la tessera a casa.

Esempi di ricerca con il metodo `findall()` di `lxml`

Namespace concatenato (a + b + c) [Non lo useremo]

```
from lxml import etree
tns = '{http://www.tei-c.org/ns/1.0}'              # for TEI XML
tree = etree.parse('in.xml')
print(tns + 'body')
body = tree.findall('.//' + tns + 'body')
print(body)
```

Namespace: dizionario

```
from lxml import etree
ns = {'t': 'http://www.tei-c.org/ns/1.0',          # for TEI XML
      'xml': 'http://www.w3.org/XML/1998/namespace',  # for attributes like xml:id
      'h': 'http://www.w3.org/1999/xhtml'}             # for (X)HTML output
tree = etree.parse('in.xml')
body = tree.findall('.//t:body', ns)
print(body)
```

Cerca tutti gli elementi

```
# L'asterisco significa "tutto". In questo caso: tutti i nomi di elementi (tagname)
from lxml import etree
ns = {'t': 'http://www.tei-c.org/ns/1.0',          # for TEI XML
      'xml': 'http://www.w3.org/XML/1998/namespace',  # for attributes like xml:id
      'h': 'http://www.w3.org/1999/xhtml'}             # for (X)HTML output
tree = etree.parse('in.xml')
print(ns['t'])
body = tree.findall('.//t:extent/*', ns)
print(body)
```

Cerca elementi con un certo attributo (senza valore)

```
from lxml import etree
ns = {'t': 'http://www.tei-c.org/ns/1.0',          # for TEI XML
      'xml': 'http://www.w3.org/XML/1998/namespace',  # for attributes like xml:id
      'h': 'http://www.w3.org/1999/xhtml'}             # for (X)HTML output
tree = etree.parse('in.xml')
print(ns['t'])
body = tree.findall('.//t:measure[@unit]', ns)
print(body)
```

Cerca elementi con attributi con un certo valore

```
from lxml import etree
ns = {'t': 'http://www.tei-c.org/ns/1.0',             # for TEI XML
        'xml': 'http://www.w3.org/XML/1998/namespace',  # for attributes like xml:id
        'h': 'http://www.w3.org/1999/xhtml'}              # for (X)HTML output
tree = etree.parse('in.xml')
print(ns['t'])
body = tree.findall('.//t:measure[@unit="words"]', ns)
print(body)
```