ΣΤΟΧΑΣΤΙΚΕΣ ΑΝΕΛΙΞΕΙΣ

Οδηγός Χρήσης του Γραφικού εργαλείου

1 Εισαγωγή

Στόχος αυτού του φυλλαδίου είναι να ξεναγηθούμε στο εργαλείο οπτικοποίησης που περιλαμβάνει η πλήρης έκδοση της βιβλιοθήκη των μαρκοβιανών αλυσίδων που σας δόθηκε.

1.1 Πριν ξεκινήσουμε

1. Στο πρώτο εργαστήριο σας δόθηκε η light έκδοση της βιβλιοθήκης σε Python, (simple_markov_chain_lib.py). Αυτή είναι και η έκδοση που συστήνουμε να χρησιμοποιείτε όταν θέλετε αποκλειστικά πειραματικά αποτελέσματα χωρίς την γραφική απεικόνιση της αλυσίδας.

2. Στόχος μας αυτή την φορά όμως είναι να παρακολουθήσουμε και γραφικά την αλυσίδα σε ένα πραγματικού-χρόνου animation. Για τον λόγο αυτό θα χρειαστεί να έχετε κατεβάσει την βιβλιοθήκη markov_chain_lib.py.

3. Υποθέτουμε ότι έχετε ολοκληρώσει επιτυχώς τα βήματα για την εγκατάσταση του προσομοιωτή ώστε να έχετε την βιβλιοθήκη σχεδιασμού γράφων και τα υπόλοιπα εργαλεία ή την χειροκίνητη εγκατάσταση των πακέτων όπως προτάθηκε στο τελευταίο μέρος του οδηγού εγκατάστασης.

2 Πρώτη επαφή

Στο πρώτο εργαστήριο, η πειραματική τακτική ήταν να πειραματίστειτε με τους κώδικες που σας δόθηκαν, να τους πειράξετε και να εκτελέσετε τους δικούς σας χρησιμοποιώντας την εντολή python my_script.py . Αυτή η τακτική όμως μας εμποδίζει να χρησιμοποίησουμε την διάδραση του διερμηνευτή της Python. Σε αυτό το μέρος του εργαστηρίου θέλουμε να πειραματιστούμε με αυτή την δυνατότητα που μας παρέχει η γλώσσα και να την εκμεταλλευτούμε για διαδραστικούς σκοπούς.

Συνεπώς:

1. Ανοίξτε ένα τερματικό.

2. Εκτελέστε την εντολή

```
python
```

3. Έχοντας δώσει σαν εντολή απλά την λέξη έχουμε ζητήσει από το λειτουργικό να ανοίξει ο φλοιός της Python, χωρίς να τρέξει κάποιο πρόγραμμα.

4. Πειραματιστείτε με τον φλοιό δίνοντας απλές εντολές όπως

```
>>3+2
5
>>print "Hello world"
Hello world
```

5. Για να κλείσετε τον φλοιό είτε πατήστε CLTRL-D είτε δώστε την εντολή

```
exit()
```

3 Ας δούμε την πρώτη μας αλυσίδα

Για να δούμε την πρώτη μας αλυσίδα θα πρέπει να έχουμε δύο βασικά αρχεία σε ένα κοινό φάκελο:

- Την βιβλιοθήκη markov_chain_lib.py
- Ένα απλό python script που ας ονομάσουμε my_markov_chain.py όπως π.χ. αυτό που κατασκευάσατε για το τένις (ή κάποια άλλη άσκηση) στο προηγούμενο εργαστήριο. Φροντίστε μόνο να εισαγάγετε την καινούργια βιβλιοθήκη αντικαθιστώντας στην αρχή του προγράμματος την εντολή

  ```
  import simple_markov_chain_lib.py as lib
  ```

  από την εντολή

  ```
  import markov_chain_lib.py as lib
  ```

  Για να τρέχει το πρόγραμμα γρήγορα ίσως θέλετε να αλλάξετε και το πλήθος των επαναλήψεων σε $N = 1$.
- Επίσης θέλουμε και το τερματικό να βρίσκεται στο φάκελο που βρίσκονται αυτά τα δύο αρχεία. Αυτό μπορείτε να το κάνετε πολύ εύκολα στο λειτουργικό σύστημα του προσομοιωτή, αφού με δεξί κλίκ στο φάκελο που βρίσκεται μπορείτε να επιλέξετε "Άνοιγμα τερματικού εδώ".

1. Αν εκτελούσαμε τώρα σαν εντολή python my_markov_chain.py, θα εκτελούσε το πρόγραμμα της αλυσίδας και θα επέστρεφε χωρίς να μπορέσουμε να πειραματιστούμε με επιπλέον εντολές.
2. Για αυτό θα εκτελέσουμε απλά την εντολή python. Αφού ανοίξει ο φλοιός των εντολών θα δώσουμε ως εντολή

   ```
   >> execfile('my_markov_chain.py')
   ```

   Ίσως χρειαστούν ένα-δύο λεπτά για αυτή την διαδικασία. Με αυτό το τρόπο θα εκτελέσει το κώδικα που μας δόθηκε αλλά θα μπορέσουμε να προσθέσουμε και άλλες εντολές χωρίς να χαθεί η μεταβλητή

   ```
   m = lib.markov_chain(init_probs,markov_table)
   ```

   που έχουμε δημιουργήσει.
3. Συνεπώς ας δοκιμάσουμε να δούμε την αλυσίδα μας εκτελώντας την εντολή:

   ```
   >> m.show()
   ```

4. Καλώς εχόντων των πραγμάτων σας έχει εμφανιστεί το παράθυρο με τον γράφο των δυνατών μεταβάσεων της μαρκοβιανής αλυσίδας.

4 First Task

4.1 Learning

- Αρχικά παρατηρήστε ότι μέχρι να κλείσετε τον γράφημα με την αλυσίδα, ο φλοιός μπλοκάρει και δεν συνεχίζει. Συνεπώς ας κλείσουμε το πρώτο αυτό στιγμιότυπο της αλυσίδας μας και ας επιστρέψουμε στην γραμμή εντολών του φλοιού.
- Ας επαναλάβουμε την εντολή:

  ```
  >> m.show()
  ```

  Προσπαθείστε να κάνετε drag τους διαφόρους κόμβους και να τους τοποθετήσετε όπου επιθυμείτε εσείς. Στην συνέχεια κλείστε το παράθυρο.

- Αν επαναλάβετε την εντολή:

  ```
  >> m.show()
  ```

  θα δείτε ότι οι κόμβοι της αλυσίδας έχουνε παραμείνει εκεί που τους τοποθετήσατε την τελευταία φορά. Αν θέλετε να επιστέψουν στην πρώτη τους θέση, κλείστε το παράθυρο και δώστε την εντολή

  ```
  >> m.show(init_positions=True)
  ```

  Επίσης αν δεν θέλετε να διατηρούνται οι αλλαγές μπορείτε να χρησιμοποιήσετε την επιλογή

  ```
  >> m.show(save_positions=False)
  ```

- Ας δοκιμάσουμε τώρα τον ακόλουθο πιο ενδιαφέρον συνδιασμό.

  ```
  >> m.show(save_positions=False,geometry=(720,720),weights=True)
  ```

  Με αυτόν τον συνδυασμό ζητάμε στην βιβλιοθήκη να μας δημιουργήσει ένα παράθυρο 720×720, να εμφανίσει τις πιθανότητες μετάβασης ως βάρη των ακμών και να μην διατηρήσει τις αλλαγές στην θέση των κόμβων.

- Προσοχή !! Κάθε φορά που κάνετε m.show(...) με οποιαδήποτε ορίσματα το πρόγραμμα σας παγώνει μέχρι να κλείσετε εσείς την εικόνα της αλυσίδας. Αφού κλείσετε το παραθυρο γραφικών είστε και πάλι ελεύθεροι να εκτελέσετε οτιδήποτε θέλετε.

- Ας ξεκινήσουμε την αλυσίδα, και ας δούμε ποιος κόμβος επιλέχθηκε ως αρχική κατάσταση

  ```
  >> m.start()
  >> m.show()
  ```

  Καλώς εχόντων των πραγμάτων, υπάρχει μια κορυφή στο γράφο που έχει την τύχη να είναι πράσινη, αφού κλείσετε το παράθυρο μπορείτε να εκτελέσετε:

  ```
  >> print m.running_state
  ```

  και να επαληθεύσετε ότι πράγματι ότι η πράσινη κορυφή είναι η παρούσα κατάσταση στην οποία βρίσκεται η αλυσίδα.

- Ας προχωρήσουμε όμως ένα βήμα την αλυσίδα. Ας τυπώσουμε την νέα κατάσταση και ας ελέγξουμε αν όντως αυτή είναι η παρούσα πράσινη κατάσταση.

  ```
  >> m.move()
  >> print m.running_state
  >> m.show()
  ```

  Επαναλάβετε τρεις-τέσσερις φορές αυτή την διαδικασία.

- Ας αποθηκεύσουμε τώρα την παρούσα κατάσταση της αλυσίδας.

  ```
  >> m.save_draw()
  >> m.move()
  >> m.save_draw()
  >> m.move()
  >> m.save_draw()
  ```

  Με αυτή την ομάδα εντολών αποθηκεύουμε τρία διαδοχικά στιγμυότυπα της αλυσίδας στον φάκελο που βρισκόμαστε σε μορφή pdf και με τα ονόματα mc_frame.pdf,mc_frame1.pdf,mc_frame2.pdf. Η αρίθμηση αυξάνει αυτόματα από το πρόγραμμα.

- Τέλος αν θέλετε λίγες περισσότερες επιλογές μπορείτε να ρυθμίσετε χειροκίνητα σε ποιο φάκελο, με τι όνομα και σε τι format και γεωμετρία επιθυμείτε την εικόνα σας.

```
m.save_draw(directory=".",choosen_format="png",filename="foo_frame",geometry=(640,640),manually=True)
```

Αυτό σημαίνει ότι:

1. Επιθυμούμε να αποθηκεύσουμε την εικόνα στον φάκελο με διεύθυνση ¨.¨, που στα Linux σημαίνει *στον φάκελο που βρισκόμαστε τώρα*.
2. Επιθυμούμε να αποθηκεύσουμε την εικόνα σε png format. Για την παρούσα έκδοση οι επιλογές σας περιορίζονται σε $\{pdf, png\}$.
3. Η εικόνα που θα αποθηκευτεί θα έχει όνομα foo_frame.png και μέγεθος 640×640

4.2 Testing

Αποθηκεύσετε σε μορφή png την κατάσταση μιας αγαπημένης αλυσίδας σας τις χρονικές στιγμές 1, 3, 5 και 7.

5 Second Task— Let's run

5.1 Learning

- Αυτή την φορά θέλουμε να δούμε την αλυσίδα να τρέχει. Συνεπώς φροντίστε η αλυσίδα σας να είναι αρκετά ¨ταξιδιάρα' και δώστε την εντολή start για να επανεκκινήσουμε την αλυσίδα και να τρέξουμε 20 βήματα της αλυσίδας

```
>> m.start()
>> m.run(20)
```

- Ας δοκιμάσουμε να κάνουμε την αλυσίδα να τρέχει λίγο πιο αργά :

```
>> m.run(10,sleeping_time=0.5)
```

ακόμα πιο αργά:

```
>> m.run(10,sleeping_time=1.5)
```

αλλά και πιο γρήγορα:

```
>> m.run(10,sleeping_time=0.25)
```

5.2 Testing

Πειραματιστείτε με την αγαπημένη σας αλυσίδα, προσαρμόζοντας τον χρόνο καθυστέρησης και τον αριθμό των βημάτων.

6 Τεχνικές απορίες

Για οποιαδήποτε τεχνική απορία, ΣΤΕΙΛΤΕ email στο stochproc.tech@gmail.com