Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο
ΥΠ23 Τεχνητή Νοημοσύνη

Tutorial: Εισαγωγή στην Python

Έκδοση 2022-1.2

Διδάσκων: Χρήστος Δίου

1 Εισαγωγή

Στο μάθημα θα χρησιμοποιήσουμε τα pacman projects που αναπτύχθηκαν στο Πανεπιστήμιο του Berkeley (http://ai.berkeley.edu) ως διδακτικό εργαλείο για τους αλγόριθμους Τεχνητής Νοημοσύνης που θα μάθετε αυτό το εξάμηνο. Οι λύσεις σας θα αναπτυχθούν σε γλώσσα Python.

Οι παρακάτω ενότητες δίνουν συνοπτικές οδηγίες για την εγκατάσταση των απαραίτητων εργαλείων ώστε να λύσετε τις ασκήσεις. Επίσης δίνονται κάποιες ασκήσεις στη γλώσσα Python καθώς και στον autograder. Οι ασκήσεις αυτές δε βαθμολογούνται ωστόσο σας προετοιμάζουν για τις επόμενες ασκήσεις του μαθήματος (που θα βαθμολογηθούν).

2 Python

2.1 Εγκατάσταση περιβάλλοντος Python

Μπορείτε να χρησιμοποιήσετε ένα δικό σας περιβάλλον (python environment) και κάποιο IDE όπως το Pycharm (https://www.jetbrains.com/pycharm/). Είναι σημαντικό να χρησιμοποιήσετε έκδοση της Python έως 3.6 (διαφορετικά δε θα λειτουργεί ο autograder). Παρακάτω δίνονται οδηγίες για την εγκατάσταση σε περιβάλλον UNIX/Linux μέσω Anaconda.

Αρχικά εγκαταστήστε το Anaconda (ή το Miniconda) για Python3 από τον παρακάτω ιστότοπο:
https://docs.anaconda.com/anaconda/install/

Έπειτα φτιάξτε ένα conda environment με την έκδοση 3.6 της Python ως εξής

```
conda create --name yp23 python=3.6
```

Μπορείτε να δώσετε ότι όνομα θέλετε στο περιβάλλον σας, εδώ δώσαμε το `yp23`.

Έπειτα, μπορείτε να ενεργοποιήσετε το περιβάλλον με την εντολή

```
conda activate yp23
```

Για να απενεργοποιήσετε το περιβάλλον μπορείτε να εκτελέσετε την εντολή

```
conda deactivate
```

Σε παλιότερες εκδόσεις της Anaconda η αντίστοιχες εντολές ήταν `source activate yp23` και `source deactivate`.

Για την εγκατάσταση κάποιου περιβάλλοντος python για το Pycharm δίνονται οδηγίες στους παρακάτω συνδέσμους.

- https://docs.anaconda.com/anaconda/user-guide/tasks/pycharm/
- https://www.jetbrains.com/help/pycharm/conda-support-creating-conda-virtual-environment.html

Επιπλέον, για εξάσκηση μπορείτε να δουλέψετε σε ένα έτοιμο colab notebook:

- https://colab.research.google.com

2.2 Εισαγωγή στην Python

Υπάρχουν πολλές πηγές για την εκμάθηση της Python online, σε περίπτωση που δε γνωρίζετε τη γλώσσα. Η επίσημη τεκμηρίωση και το tutorial είναι αρκετά καλά:

https://docs.python.org/3/tutorial/

Επίσης πολύ ενδιαφέρον είναι και το υλικό του μαθήματος "Scientific Python" από το Πανεπιστήμιο του Stanford, το οποίο εκτός από βασικά στοιχεία της γλώσσας Python, σας προσφέρει και μία εισαγωγή στις βιβλιοθήκες Numpy, Pandas και scikit-learn.

http://web.stanford.edu/class/cme193/syllabus.html

Για τις ανάγκες του μαθήματος, θα σας δοθούν (μέσω του eclass) ορισμένα notebooks βασισμένα στο μάθημα CME-193 του Stanford. Ωστόσο αξίζει να προσπαθήσετε να υλοποιήσετε τις λύσεις και τοπικά στον υπολογιστή σας.

Η παρακάτω εισαγωγή βασίζεται σε μία σύντομη εισαγωγή στην python από το πανεπιστήμιο του Berkeley.

2.3 Πως καλούμε την python

Η python μπορεί να λειτουργήσει διαδραστικά ή να εκτελεστεί από τη γραμμή εντολών. Για παράδειγμα στον υπολογιστή μου,

```
diou@clio:~$ conda activate tutorial
(tutorial) diou@clio:~$ python
Python 3.6.13 |Anaconda, Inc.| (default, Jun  4 2021, 14:25:59)
[GCC 7.5.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Πλέον μπορούμε να εκτελούμε εντολές της python διαδραστικά. Εναλλακτικά μπορούμε να γράψουμε τις εντολές σε ένα αρχείο που ονομάζεται "script" και να τις εκτελούμε όλες μαζί.

Για παράδειγμα ετοιμάστε ένα αρχείο `hello.py` με τα παρακάτω περιεχόμενα:

```
s = "hello, world!"
su = s.upper()
print(su)
```

και εκτελέστε το ως εξής

```
(tutorial) diou@clio:~$ python hello.py
HELLO, WORLD!
```

όπου υποθέτουμε ότι βρίσκεστε στον ίδιο κατάλογο με το αρχείο.

Τα παραπάνω παραδείγματα είναι σε λειτουργικό unix, αλλά το ίδιο εφαρμόζεται και σε Windows, είτε από τη γραμμή εντολών ή από την κονσόλα του IDE.

Εκτελέστε τα παρακάτω διαδραστικά (μπορείτε επίσης να εκτελείτε και κώδικα που θα δείτε στα notebooks)

2.4 Τελεστές

Αριθμητικοί τελεστές

```
>>> 1 + 1
2
>>> 2 * 3
6
```

Λογικοί τελεστές (υπάρχει τιμή `True` και `False`)

```
>>> 1 == 0
False
>>> not (1 == 0)
True
>>> (2 == 2) and (2 == 3)
False
>>> (2 == 2) or (2 == 3)
True
```

2.5 Συμβολοσειρές

Ο τελεστής `+`, όταν χρησιμοποιείται με συμβολοσειρές είναι ο τελεστής συνένωσης (concatenation)

```
>>> 'deep' + "learning''
'deeplearning'
```

Υπάρχουν αρκετές μέθοδοι επεξεργασίας συμβολοσειρών ενσωματωμένες στην python.

```
>>> 'artificial'.upper()
'ARTIFICIAL'
>>> 'HELP'.lower()
'help'
>>> len('Help')
4
```

Μπορούμε να χρησιμοποιήσουμε είτε μονά `' '` ή διπλά `" "` εισαγωγικά για να ορίσουμε συμβολοσειρές.

Μπορούμε επίσης να αποθηκεύσουμε εκφράσεις σε μεταβλητές.

```
>>> s = 'hello world'
>>> print(s)
hello world
>>> s.upper()
'HELLO WORLD'
>>> len(s.upper())
11
>>> num = 8.0
>>> num += 2.5
>>> print(num)
10.5
```

Χρησιμοποιώντας την εντολή `dir` μπορούμε να δούμε όλες τις μεθόδους που είναι διαθέσιμες για ένα αντικείμενο, ενώ με την `help` μπορούμε να διαβάσουμε την τεκμηρίωση για κάποια μέθοδο. Για παράδειγμα για να δούμε τι μεθόδους έχουμε διαθέσιμες για συμβολοσειρές:

```
>>> s = 'abc'

>>> dir(s) ['__add__', '__class__', '__contains__', '__delattr__', '__doc__',
  '__eq__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__',
  '__getslice__', '__gt__', '__hash__', '__init__', '__le__', '__len__',
  '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__',
  '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__',
  '__str__', 'capitalize', 'center', 'count', 'decode', 'encode', 'endswith',
  'expandtabs', 'find', 'index', 'isalnum', 'isalpha', 'isdigit', 'islower',
  'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip',
  'replace', 'rfind', 'rindex', 'rjust', 'rsplit', 'rstrip', 'split',
  'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate',
  'upper', 'zfill']
```

```
>>> help(s.find)
Help on built-in function find:

find(...) method of builtins.str instance
    S.find(sub[, start[, end]]) -> int

    Return the lowest index in S where substring sub is found,
    such that sub is contained within S[start:end].  Optional
    arguments start and end are interpreted as in slice notation.

    Return -1 on failure.


>>> s.find('b')
1
```

Οι μέθοδοι που έχουν κάτω παύλα (underscore) είναι βοηθητικές και συνήθως δεν τις χρησιμοποιούμε. Πατώντας 'q' βγαίνουμε από την οθόνη βοήθειας.

2.6 Λίστες

Οι λίστες αποθηκεύουν μία ακολουθία αντικειμένων, τα οποία μπορεί να μεταβληθούν.

```
>>> fruits = ['apple', 'orange', 'pear', 'banana']
>>> fruits[0]
'apple'
```

Μπορούμε να συνενώσουμε λίστες με τον τελεστή '+'.

```
>>> otherFruits = ['kiwi', 'strawberry']
>>> fruits + otherFruits
>>> ['apple', 'orange', 'pear', 'banana', 'kiwi', 'strawberry']
```

Η python επίσης επιτρέπει αρνητικούς δείκτες (ξεκινούν από το τέλος της λίστας). Για παράδειγμα `fruits[-1]` αντιστοιχεί στο τελευταίο στοιχείο της λίστας `fruits` (που είναι 'banana').

```
>>> fruits[-2]
'pear'
>>> fruits.pop()
'banana'
>>> fruits
['apple', 'orange', 'pear']
>>> fruits.append('grapefruit')
>>> fruits
['apple', 'orange', 'pear', 'grapefruit']
>>> fruits[-1] = 'pineapple'
>>> fruits
['apple', 'orange', 'pear', 'pineapple']
```

Μπορούμε επίσης να προσπελάσουμε πολλά διαδοχικά στοιχεία χρησιμοποιώντας τον τελεστή ':'. Για παράδειγμα η έκφραση `fruits[1:3]` επιστρέφει τα αντικείμενα στις θέσεις 1 και 2 (η αρίθμηση ξεκινά από το μηδέν). Γενικά η έκφραση `fruits[start:stop]` επιστρέφει τα στοιχεία start, start+1, ..., stop-1. Μπορούμε επίσης να γράψουμε `fruits[start:]` που επιστρέφει όλα τα στοιχεία από το start και έπειτα, ενώ το `fruits[:end]` επιστρέφει όλα τα στοιχεία πριν τη θέση end.

```
>>> fruits[0:2]
['apple', 'orange']
>>> fruits[:3]
['apple', 'orange', 'pear']
```

```
>>> fruits[2:]
['pear', 'pineapple']
>>> len(fruits)
4
```

Η λίστα μπορεί να περιέχει στοιχεία οποιουδήποτε τύπου. Π.χ. μπορεί να έχουμε λίστες από λίστες.

```
>>> lstOfLsts = [['a', 'b', 'c'], [1, 2, 3], ['one', 'two', 'three']]
>>> lstOfLsts[1][2]
3
>>> lstOfLsts[0].pop()
'c'
>>> lstOfLsts
[['a', 'b'], [1, 2, 3], ['one', 'two', 'three']]
```

Επεξεργαστείτε ορισμένες συναρτήσεις από λίστες

```
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__',
'__delslice__', '__doc__', '__eq__', '__ge__', '__getattribute__',
'__getitem__', '__getslice__', '__gt__', '__hash__', '__iadd__', '__imul__',
'__init__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__',
'__rmul__', '__setattr__', '__setitem__', '__setslice__', '__str__',
'append', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse',
'sort']

>>> help(list.reverse)
          Help on built-in function reverse:

          reverse(...)
          L.reverse() -- reverse \*IN PLACE\*


>>> lst = ['a', 'b', 'c']
>>> lst.reverse()
>>> ['c', 'b', 'a']`


\subsection{Tuples}

Τα tuples (ν-άδες) είναι δομή δεδομένων παρόμοια με τη λίστα, με τη διαφορά ότι
τα περιεχόμενά τους δεν αλλάζουν αφότου δημιουργηθούν. Ορίζονται με παρενθέσεις
αντί για τετράγωνες αγκύλες.

\begin{minted}{ipython3}
>>> pair = (3, 5)
>>> pair[0]
3
>>> x, y = pair
>>> x
3
>>> y
5
>>> pair[1] = 6
TypeError: object does not support item assignment
```

2.7 Σύνολα (Sets)

Ένα σύνολο (set) είναι άλλη μία δομή δεδομένων, στην οποία όμως τα στοιχεία δεν έχουν κάποια σειρά και δεν εμφανίζονται πάνω από μία φορά.

```
>>> shapes = ['circle', 'square', 'triangle', 'circle']
>>> setOfShapes = set(shapes)
```

Εναλλακτικός τρόπος:

```
  >>> setOfShapes = {‘circle’, ‘square’, ‘triangle’, ‘circle’}
```

Μπορούμε να προσθέτουμε αντικείμενα στο σύνολο, καθώς και να κάνουμε πράξεις συνόλων (ένωση, τομή, διαφορά)

```
>>> setOfShapes
set(['circle', 'square', 'triangle'])
>>> setOfShapes.add('polygon')
>>> setOfShapes
set(['circle', 'square', 'triangle', 'polygon'])
>>> 'circle' in setOfShapes
True
>>> 'rhombus' in setOfShapes
False
>>> favoriteShapes = ['circle', 'triangle', 'hexagon']
>>> setOfFavoriteShapes = set(favoriteShapes)
>>> setOfShapes - setOfFavoriteShapes
set(['square', 'polygon'])
>>> setOfShapes & setOfFavoriteShapes
set(['circle', 'triangle'])
>>> setOfShapes | setOfFavoriteShapes
set(['circle', 'square', 'triangle', 'polygon', 'hexagon'])
```

Σημειώστε ότι τα αντικείμενα του συνόλου δεν έχουν συγκεκριμένη σειρά.

2.8 Λεξικά (dictionaries)

Μία άλλη πολύ χρήσιμη δομή δεδομένων της python είναι το dictionary. Αυτό αποθηκεύει μία απεικόνιση από έναν τύπο αντικειμένου (key) σε έναν άλλο (value). Το κλειδί (key) πρέπει να είναι κάποιος μη μεταβλητός (immutable) τύπος (string, number, tuple). Η τιμή (value) μπορεί να είναι οποιοσδήποτε τύπος.

Παράδειγμα:

```
>>> studentIds = {'knuth': 42.0, 'turing': 56.0, 'nash': 92.0}
>>> studentIds['turing']
56.0
>>> studentIds['nash'] = 'ninety-two'
>>> studentIds
{'knuth': 42.0, 'turing': 56.0, 'nash': 'ninety-two'}
>>> del studentIds['knuth']
>>> studentIds
{'turing': 56.0, 'nash': 'ninety-two'}
>>> studentIds['knuth'] = [42.0, 'forty-two']
>>> studentIds
{'knuth': [42.0, 'forty-two'], 'turing': 56.0, 'nash': 'ninety-two'}
>>> studentIds.keys()
['knuth', 'turing', 'nash']
>>> studentIds.values()
[[42.0, 'forty-two'], 56.0, 'ninety-two']
>>> studentIds.items()
[('knuth', [42.0, 'forty-two']), ('turing',56.0), ('nash', 'ninety-two')]
```

```
>>> len(studentIds)
3
```

Χρησιμοποιήστε τις εντολές `dir` και `help` ώστε να μάθετε περισσότερα για τα dictionaries.

2.9 Scripts

Για να δείτε πως μπορείτε να χρησιμοποιείτε scripts στην python γράψτε τον παρακάτω κώδικα σε ένα αρχείο `foreach.py`:

```
# This is what a comment looks like
fruits = ['apples', 'oranges', 'pears', 'bananas']
for fruit in fruits:
    print(fruit + ' for sale')

fruitPrices = {'apples': 2.00, 'oranges': 1.50, 'pears': 1.75}
for fruit, price in fruitPrices.items():
    if price < 2.00:
        print('%s cost %f a pound' % (fruit, price))
    else:
        print(fruit + ' are too expensive!')
```

Αν το τρέξουμε όπως προηγουμένως:

```
(tutorial) diou@clio:~$ python foreach.py
apples for sale
oranges for sale
pears for sale
bananas for sale
apples are too expensive!
oranges cost 1.500000 a pound
pears cost 1.750000 a pound
```

Άλλες χρήσιμες ενσωματωμένες συναρτήσεις είναι οι `map` και `filter`:

```
>>> list(map(lambda x: x * x, [1, 2, 3]))
[1, 4, 9]
>>> list(filter(lambda x: x > 3, [1, 2, 3, 4, 5, 4, 3, 2, 1]))
[4, 5, 4]
```

Παρακάτω βλέπουμε τη λειτουργία “comprehension” της Python:

```
nums = [1, 2, 3, 4, 5, 6]
plusOneNums = [x + 1 for x in nums]
oddNums = [x for x in nums if x % 2 == 1]
print(oddNums)
oddNumsPlusOne = [x + 1 for x in nums if x % 2 == 1]
print(oddNumsPlusOne)
```

Αν τρέξουμε τον κώδικα παράγει

```
(tutorial) diou@clio:~$ python listcomp.py
[1, 3, 5]
[2, 4, 6]
```

2.10 Στοίχιση και κενά

Σε αντίθεση με τις περισσότερες γλώσσες προγραμματισμού τα μπλοκ του κώδικα στην python ορίζονται μέσω της στοίχισης. Για παράδειγμα ο παρακάτω κώδικας

```
if 0 == 1:
    print('We are in a world of arithmetic pain')
print('Thank you for playing')
```

θα τυπώσει “Thank you for playing”. Αν όμως είχαμε γράψει

```
if 0 == 1:
    print('We are in a world of arithmetic pain')
    print('Thank you for playing')
```

δε θα είχαμε καθόλου έξοδο.

2.11 Συναρτήσεις

Στην python μπορούμε να ορίζουμε συναρτήσεις:

```
fruitPrices = {'apples': 2.00, 'oranges': 1.50, 'pears': 1.75}

def buyFruit(fruit, numPounds):
    if fruit not in fruitPrices:
        print("Sorry we don't have %s" % (fruit))
    else:
        cost = fruitPrices[fruit] * numPounds
        print("That'll be %f please" % (cost))

# Main Function
if __name__ == '__main__':
    buyFruit('apples', 2.4)
    buyFruit('coconuts', 2)
```

Ο έλεγχος `__name__ == '__main__'` ικανοποιείται μόνο όταν καλούμε απευθείας το αρχείο από τη γραμμή εντολών και δεν το εισάγουμε, π.χ. με `import`.

2.12 Αντικείμενα

Στην python μπορούμε να ορίζουμε κλάσεις αντικειμένων. Το παρακάτω παράδειγμα ορίζει μία κλάση που ονομάζεται `FruitShop`:

```
class FruitShop:

    def __init__(self, name, fruitPrices):
        """
            name: Name of the fruit shop

            fruitPrices: Dictionary with keys as fruit
            strings and prices for values e.g.
            {'apples': 2.00, 'oranges': 1.50, 'pears': 1.75}
        """
        self.fruitPrices = fruitPrices
        self.name = name
        print('Welcome to %s fruit shop' % (name))

    def getCostPerPound(self, fruit):
        """
            fruit: Fruit string
        Returns cost of 'fruit', assuming 'fruit'
        is in our inventory or None otherwise
        """
        if fruit not in self.fruitPrices:
```

```
            return None
        return self.fruitPrices[fruit]

    def getPriceOfOrder(self, orderList):
        """
            orderList: List of (fruit, numPounds) tuples

        Returns cost of orderList, only including the values of
        fruits that this fruit shop has.
        """
        totalCost = 0.0
        for fruit, numPounds in orderList:
            costPerPound = self.getCostPerPound(fruit)
            if costPerPound != None:
                totalCost += numPounds * costPerPound
        return totalCost

    def getName(self):
        return self.name
```

Ετοιμάζουμε ένα αρχείο `shop.py` που περιέχει τον παραπάνω ορισμό της κλάσης. Γράφοντας `import shop` μπορούμε να χρησιμοποιούμε αυτή την κλάση σε άλλα αρχεία κώδικα python. Παράδειγμα:

```
import shop

shopName = 'the Berkeley Bowl'
fruitPrices = {'apples': 1.00, 'oranges': 1.50, 'pears': 1.75}
berkeleyShop = shop.FruitShop(shopName, fruitPrices)
applePrice = berkeleyShop.getCostPerPound('apples')
print(applePrice)
print('Apples cost $%.2f at %s.' % (applePrice, shopName))

otherName = 'the Stanford Mall'
otherFruitPrices = {'kiwis': 6.00, 'apples': 4.50, 'peaches': 8.75}
otherFruitShop = shop.FruitShop(otherName, otherFruitPrices)
otherPrice = otherFruitShop.getCostPerPound('apples')
print(otherPrice)
print('Apples cost $%.2f at %s.' % (otherPrice, otherName))
print("My, that's expensive!")
```

Αν ο κώδικας αυτός είναι στο αρχείο `shopTest.py` μπορούμε να τον τρέξουμε ως εξής:

```
$ python shopTest.py
Welcome to the Berkeley Bowl fruit shop
1.0
Apples cost $1.00 at the Berkeley Bowl.
Welcome to the Stanford Mall fruit shop
4.5
Apples cost $4.50 at the Stanford Mall.
My, that's expensive!
```

Έχοντας εισάγει τον ορισμό της κλάσης FruitShop μέσω του import, η γραμμή

```
berkeleyShop = shop.FruitShop(shopName, fruitPrices)
```

δημιουργεί ένα αντικείμενο τύπου `FruitShop`, καλώντας τη συνάρτηση `__init__`. Σημειώστε ότι οι μέθοδοι που ανήκουν σε κάποια κλάση έχουν ως πρώτο όρισμα το `self`, που είναι το ίδιο το αντικείμενο.

2.13 Μεταβλητές της κλάσης και του αντικειμένου

Μία μεταβλητή μπορεί να είναι η ίδια για την κλάση (δηλ για όλα τα αντικείμενα) η να ορίζεται για κάθε αντικείμενο. Για παράδειγμα έστω αρχείο `person_class.py` με τα παρακάτω περιεχόμενα

```
class Person:
    population = 0

    def __init__(self, myAge):
        self.age = myAge
        Person.population += 1

    def get_population(self):
        return Person.population

    def get_age(self):
        return self.age
```

Αρχικά το τρέχουμε ώστε να μεταγλωττιστεί

```
$ python person_class.py
```

Έπειτα χρησιμοποιούμε την κλάση:

```
>>> import person_class
>>> p1 = person_class.Person(12)
>>> p1.get_population()
1
>>> p2 = person_class.Person(63)
>>> p1.get_population()
2
>>> p2.get_population()
2
>>> p1.get_age()
12
>>> p2.get_age()
63
```

Η μεταβλητή `age` είναι μεταβλητή αντικειμένου (instance variable) ενώ η μεταβλητή `population` είναι μεταβλητής της κλάσης (στατική μεταβλητή, ίδια για όλα τα αντικείμενα τύπου Person).

2.14 Ασκήσεις

Άσκηση 1: Λίστες και λεξικά

Χρησιμοποιώντας την εντολή `dir` μπορείτε να βλέπετε τα περιεχόμενα ενός αντικειμένου στην Python, ενώ με την `help` μπορείτε να εκτυπώνετε τη "βοήθεια" για ένα αντικείμενο της Python. Για παράδειγμα:

```
In [4]: dir(list)
Out[4]:
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
```

```
 '__getitem__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']

In [5]: help(list.pop)
Help on method_descriptor:

pop(self, index=-1, /)
    Remove and return item at index (default last).

    Raises IndexError if list is empty or index is out of range.
```

Χρησιμοποιείστε τις `dir` και `help` για να εκτυπώσετε τη βοήθεια για διαφορετικές συναρτήσεις για λίστες και λεξικά της Python.

Άσκηση 2: List comprehensions

Γράψτε ένα list comprehension που από μία λίστα συμβολοσειρών δημιουργεί μία εκδοχή με πεζά γράμματα της κάθε συμβολοσειράς που έχει μήκος μεγαλύτερο του 5.

Άσκηση 3: Quicksort

Υλοποιήστε μία συνάρτηση `quicksort()` που υλοποιεί τον αλγόριθμο quicksort χρησιμοποιώντας list comprehensions. Ο quicksort εκτελεί αναδρομικά τα ακόλουθα βήματα

1. Αν η λίστα είναι άδεια, επέστρεψε. Αλλιώς διάλεξε ένα στοιχείο της λίστας ως pivot.
2. Χώρισε τα στοιχεία σε δύο υπο-λίστες, αυτά που είναι μικρότερα του pivot και αυτά που είναι μεγαλύτερα.

3. Τρέξε αναδρομικά τη συνάρτηση για κάθε υπο-λίστα

Για περισσότερες πληροφορίες, μπορείτε να διαβάσετε γι τον quicksort στη wikipedia
https://en.wikipedia.org/wiki/Quicksort

3 Tutorial

Σας δίνεται ένα αρχείο `tutorial.zip` το οποίο περιέχει τρία ερωτήματα.

```
[diou@zenbook:../prep]$ unzip tutorial.zip
[diou@zenbook:../prep]$ cd tutorial
[diou@zenbook:../tutorial]$ ls
addition.py         __pycache__                test_cases      tutorialTestClasses.py
autograder.py       shopAroundTown.py          testClasses.py  tutorial.token
buyLotsOfFruit.py   shop.py                    testParser.py   util.py
grading.py          shopSmart.py               textDisplay.py
projectParams.py    submission_autograder.py   town.py
[diou@zenbook:../tutorial]$
```

Θα χρειαστεί να δουλέψετε με τα παρακάτω αρχεία

- `addition.py` : Άσκηση t.1
- `buyLotsOfFruit.py` : Άσκηση t.2
- `shopSmart.py` : Άσκηση t.3
- `autograder.py` : Script του autograder

Εκτελώντας τον autograder προκύπτει αυτόματα η βαθμολόγηση των απαντήσεών σας:

```
(yp23) [diou@zenbook:../tutorial]$ python autograder.py
```

Παρατηρήστε την έξοδο της παραπάνω εντολής (προφανώς ακόμα δεν έχετε λύσει τις ασκήσεις, οπότε ο βαθμός σας είναι μηδέν).

3.1 Άσκηση t.1: Πρόσθεση

Γράψτε μία συνάρτηση που επιστρέφει το άθροισμα δύο αριθμών τροποποιώντας το αρχείο `addition.py`. Μόλις την υλοποιήσετε, εκτελέστε ξανά τον autograder και δείτε αν η λύση σας λαμβάνεται ως σωστή.

3.2 Άσκηση t.2: Συνάρτηση `buyLotsOfFruit`

Προσθέστε μία συνάρτηση `buyLotsOfFruit(orderList)` στο αρχείο `buyLotsOfFruit.py` που δέχεται μία λίστα από ζευγάρια `(fruit, kg)` και επιστρέφει το κόστος της λίστας. Αν υπάρχουν φρούτα που δεν υπάρχουν στη `fruitPrices` πρέπει να εκτυπώνει ένα μήνυμα σφάλματος και να επιστρέφει `None`. Μην αλλάξετε τη μεταβλητή `fruitPrices`.

Εκτελέστε τον autograder έως ότου το ερώτημα 2 να περνά όλα τα τεστ.

3.3 Άσκηση t.3: Συνάρτηση `shopSmart`

Γράψτε τη συνάρτηση `shopSmart(orders, shops)` στο αρχείο shopSmart.py, η οποία παίρνει στην είσοδο μία `orderList` και μία λίστα από `FruitShop` και επιστρέφει το `FruitShop` που η παραγγελία σας κοστίζει λιγότερο. Μην αλλάξετε το όνομα του αρχείου ή τις μεταβλητές.

Εκτελέστε τον autograder έως ότου η συνάρτησή σας να περνά όλα τα τεστ.