Οδηγός Εργασίας

Project Python Scrabble

ΕΡΓΑΣΙΑ Θ2

ΘΕΩΡΙΕΣ ΜΑΘΗΣΗΣ & ΕΚΠΑΙΔΕΥΤΙΚΟ ΛΟΓΙΣΜΙΚΟ 2019

ΤΜ. ΠΛΗΡΟΦΟΡΙΚΗΣ ΑΠΘ ΣΤ. ΔΗΜΗΤΡΙΑΔΗΣ

Περιεχόμενα

(1) Γενική περιγραφή

(1) Τι θα χρειαστείτε για την εργασία σας

- Για να ολοκληρώσετε σωστά την εργασία σας θα χρειαστείτε:
- (α) Αυτόν τον **Οδηγό Εργασίας 2019** (που προφανώς τον έχετε ήδη)
- (β) Τα **Πακέτα διαφανειών** για την Python
- (γ) Το αρχείο **greek7.txt** που περιέχει τις λέξεις της ελληνικής γλώσσας που έχουν μέχρι και 7 γράμματα.
- *Όλα τα παραπάνω βρίσκονται στη σελίδα του μαθήματος*

- **Πηγές στο διαδίκτυο:**
- Σκράμπλ – Βικιπαίδεια: Τι είναι το Scrabble και γενικοί κανόνες του παιχνιδιού
- Python **itertools** (ενσωματωμένη βιβλιοθήκη) – ειδικά τη συνάρτηση itertools.permutations()
- Built in types (ειδικά τις μεθόδους συμβολοσειρών: strip, split)

(2) Γενική περιγραφή της εργασίας 1/2

- Ο γενικός στόχος της εργασίας είναι να αναπτύξετε μια εφαρμογή σε κώδικα Python η οποία να **υλοποιεί μια απλοποιημένη εκδοχή του παιχνιδιού Scrabble στον υπολογιστή**
- Στην εκδοχή αυτή **ένας παίκτης παίζει Scrabble με αντίπαλο τον υπολογιστή** προσπαθώντας ο καθένας να δημιουργήσει μια λέξη με υψηλή βαθμολογία με βάση τα γράμματα που έχει στη διάθεσή του σε κάθε του κίνηση.
- Ο ειδικότερος στόχος της εργασίας είναι να **γραφεί ένας αλγόριθμος που να καθοδηγεί τον υπολογιστή (ή και τον παίκτη) να παίξει το παιχνίδι**

(2) Γενική περιγραφή της εργασίας 2/2

- Η εφαρμογή σας θα πρέπει να ακολουθεί σε γενικές γραμμές την ανάλυση που περιγράφεται στις επόμενες διαφάνειες
- Μπορείτε να επιλέξετε αν θα εργαστείτε **ατομικά ή ομαδικά** (N_{max} = 2, δηλ. μέχρι 2 άτομα στην ομάδα)

- Σε περίπτωση ομαδικής εργασίας τότε **κάθε μέλος της ομάδας**:
- Α) **Γράφει και έναν δικό του/της αλγόριθμο** που ενσωματώνεται στο παιχνίδι ως επιλογή παιχνιδιού (δηλ. θα μπορεί να επιλέγεται από τις 'Ρυθμίσεις')
- Β) Θα πρέπει **να γνωρίζει την τεκμηρίωση** ολόκληρου του κώδικα που αποτελεί την εργασία

- Κάθε **βελτίωση ή επέκταση** του Σεναρίου / Αλγορίθμου του Παιχνιδιού (σε σχέση με αυτά που προτείνω στον παρόντα Οδηγό) είναι απολύτως δεκτή αρκεί να είναι ικανοποιητικά **τεκμηριωμένη** και να μου **κοινοποιηθεί έγκαιρα** από την ομάδα ώστε να πάρει την έγκρισή μου.

Περιβάλλον

- Η εφαρμογή θα τρέχει στο **περιβάλλον IDLE** της βασικής Python και θα βασίζεται σε κώδικα Python έκδοσης 3.4 ή μεγαλύτερης
- Όλες οι πληροφορίες κατά την εξέλιξη του παιχνιδιού θα εμφανίζονται στη γραμμή εντολών '>>>' **(character mode)** του περιβάλλοντος IDLE (δηλ. η εφαρμογή δεν θα είναι παραθυρική και δεν θα χρησιμοποιεί γραφικά)
- Πχ. μια πιθανή εμφάνιση μηνυμάτων από το παιχνίδι μπορεί να είναι όπως παρακάτω:

```
>>>
Στο σακουλάκι: 75 γράμματα - Παίζεις:
Διαθέσιμα Γράμματα: Ε,1 - Β,8 - Α,1 - Χ,8 - Μ,3 - Η,1 - Θ,10
Λέξη:
```

(2) Κλάσεις

Οι 5 κλάσεις του παιχνιδιού

- Προγραμματιστικά η εφαρμογή θα πρέπει:
- 1) Να χρησιμοποιεί **τουλάχιστον τις παρακάτω 5 Κλάσεις:**
- → **SakClass**: κλάση που περιγράφει αντικείμενα που λειτουργούν ως «σακουλάκι με τα γράμματα του παιχνιδιού»
- → **Player**: βασική κλάση από την οποία παράγονται οι Human και Computer
  - → **Human**: κλάση παράγωγος της Player που περιγράφει το πώς παίζει ο άνθρωπος-παίκτης
  - → **Computer**: κλάση παράγωγος της Player που περιγράφει το πώς παίζει ο υπολογιστής-παίκτης
- → **Game**: κλάση που περιγράφει το πώς εξελίσσεται μια παρτίδα (game)
- Πέρα από τις παραπάνω μπορείτε να υλοποιήσετε και άλλες κλάσεις αν θέλετε εσείς
- Περισσότερες πληροφορίες για τις κλάσεις δίνονται στις επόμενες διαφάνειες

Κλάση: **SakClass**

sak
(αντικείμενο
τύπου SakClass)

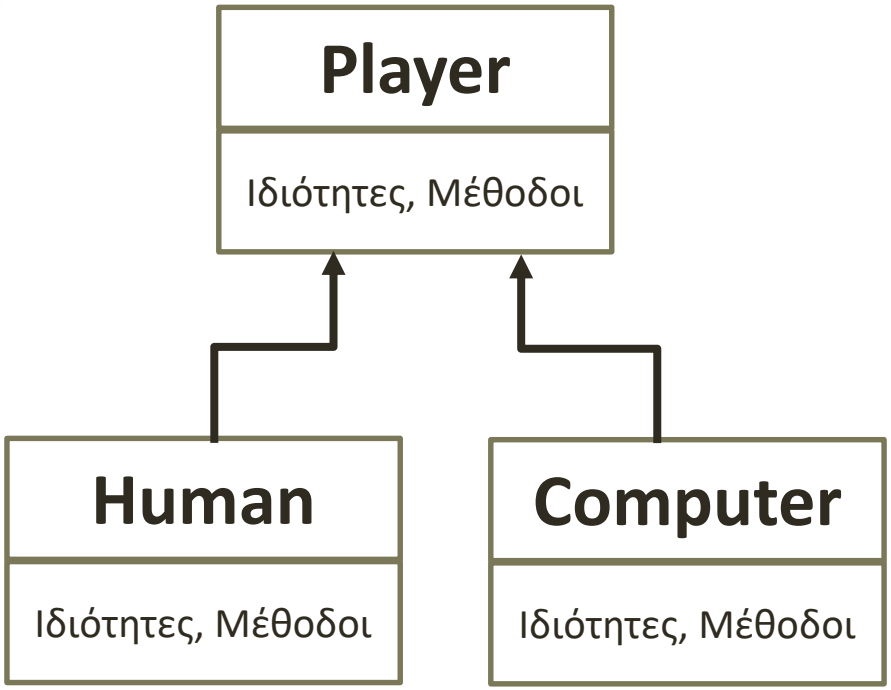
getletters()
(βγάζει από το σακουλάκι για τον
παίκτη Ν γράμματα)

ΠΑΙΚΤΗΣ

putbackletters()
(επιστρέφει γράμματα παίκτη
στο σακουλάκι)

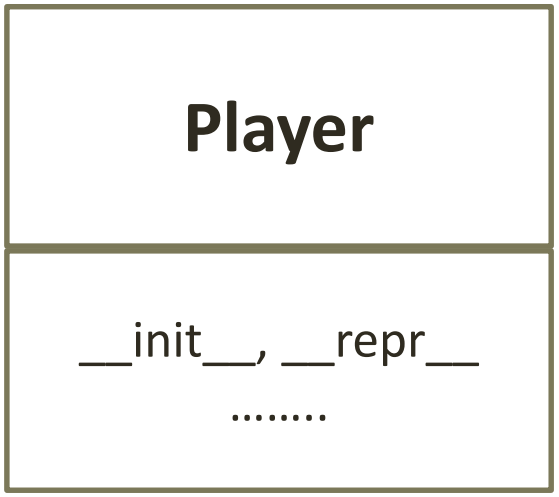
randomize_sak()
(«ετοιμάζει» το σακουλάκι
με τα γράμματα)

Κλάσεις: Player, Human & Computer



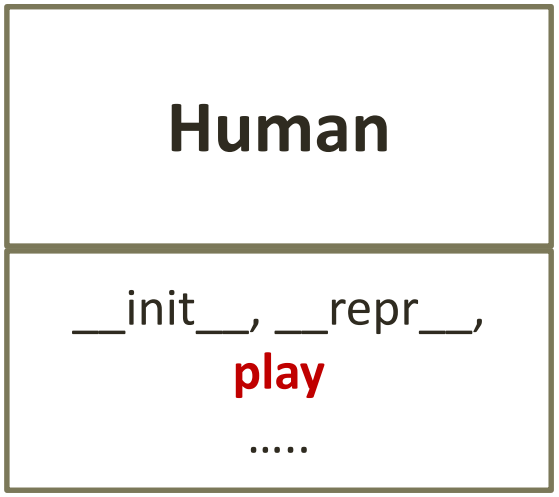
- Player: Βασική
- Human, Computer: παράγωγοι της Player

Κλάση: **Player**



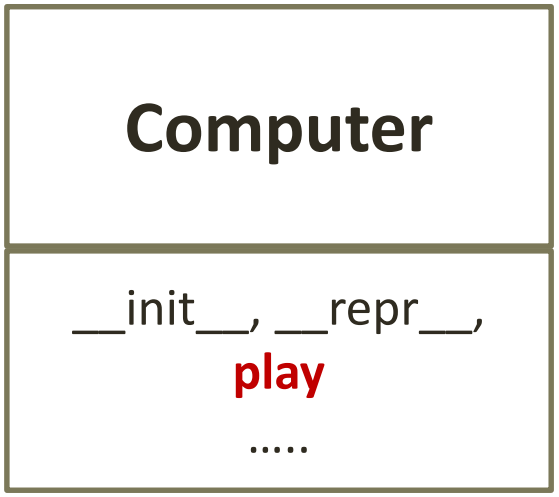
- **Player**: Βασική
- Μέθοδοι: init, repr και όποιες άλλες κρίνετε εσείς

Κλάση: **Human**



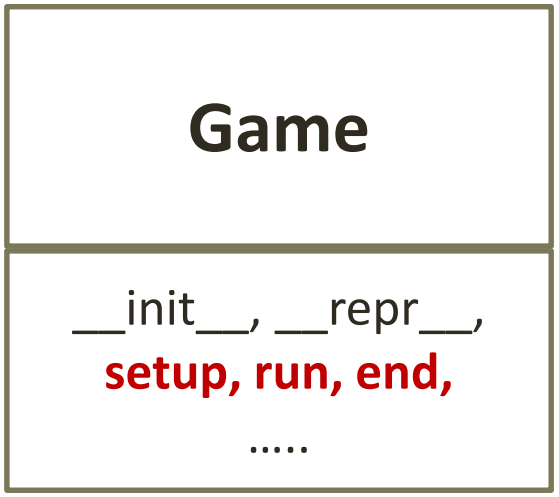
- **Player**: Βασική
- Μέθοδοι: init, repr, **play** και όποιες άλλες κρίνετε εσείς
- Η μέθοδος **play** θα υλοποιεί/καλεί τον αλγόριθμο σύμφωνα με τον οποίο θα παίζει ο παίκτης

Κλάση: **Computer**



- **Player**: Βασική
- Μέθοδοι: init, repr, **play** και όποιες άλλες κρίνετε εσείς
- Η μέθοδος **play** θα υλοποιεί/καλεί τον αλγόριθμο σύμφωνα με τον οποίο θα παίζει ο υπολογιστής

Κλάση: **Game**



- **Player**: Βασική
- Μέθοδοι: init, repr, **setup, run, end** και όποιες άλλες κρίνετε εσείς
- Η μέθοδος **setup** κάνει τις απαραίτητες ενέργειες κατά το ξεκίνημα του παιχνιδιού
- Η μέθοδος **run** 'τρέχει' το παιχνίδι
- Η μέθοδος **end** κάνει τις απαραίτητες ενέργειες κατά το κλείσιμο του παιχνιδιού

Πώς θα αποθηκεύσετε τον κώδικα των κλάσεων και του κύριου προγράμματος

- Ο κώδικας των κλάσεων θα βρίσκεται σε **ένα ξεχωριστό αρχείο** το οποίο θα ονομάσετε **classes.py**
- Ο κώδικας του κύριο προγράμματος θα βρίσκεται σε ένα αρχείο με όνομα **main-AEM.py**
  - Όπου ΑΕΜ θα γράψετε τον αριθμό του ΑΕΜ σας, πχ. main-1234.py
- Αν εργάζεστε σε ομάδα το όνομα αρχείου θα περιλαμβάνει και τα δύο ΑΕΜ το ένα μετά το άλλο **main-AEM1AEM2.py**
  - Πχ. main-12343001.py
- Ο κώδικας των κλάσεων θα εισάγεται στο κύριο πρόγραμμα με εντολή **import**

(3) Σενάριο Παιχνιδιού

Σενάριο Παιχνιδιού


```
>>>
***** SCRABBLE *****
--------------------
1: Σκορ
2: Ρυθμίσεις
3: Παιχνίδι
q: Έξοδος
--------------------
```

- **1)** Το παιχνίδι μπορεί να ξεκινά με μία εισαγωγική οθόνη που παρουσιάζει κατάλογο επιλογών στον παίκτη. Παράδειγμα μιας τέτοιας οθόνης βλέπετε δίπλα:

- Επίσης αρχικά το πρόγραμμα θα πρέπει να δημιουργεί ένα **αντικείμενο –στιγμιότυπο** (πχ. **sak**) τύπου **SakClass**
- Το αντικείμενο θα υλοποιεί το «**σακουλάκι**» του φυσικού παιχνιδιού. Δηλαδή, θα είναι δομή δεδομένων η οποία περιέχει τα γράμματα και τις αξίες τους («βαθμοί» που δίνει το κάθε γράμμα) και τις μεθόδους που σχετίζονται με το «σακουλάκι»
- Πληροφορίες για το πόσα είναι αυτά τα γράμματα υπάρχουν εδώ

sak
(αντικείμενο τύπου SakClass)

Σενάριο Παιχνιδιού 2/7

- **2)** Η εξέλιξη του παιχνιδιού έχει ως εξής:
- Το πρόγραμμα κληρώνει 7 γράμματα από το «σακουλάκι» για τον παίκτη και 7 για τον υπολογιστή. Παρουσιάζει στον παίκτη τα γράμματά του μαζί με την αξία τους (η αξία μπορεί να είναι ένας ακέραιος αριθμός που εμφανίζεται δίπλα στο γράμμα). Ταυτόχρονα το πρόγραμμα θα αφαιρεί από το «σακουλάκι» τα γράμματα που κληρώθηκαν. Η πληροφορία για το πόσα γράμματα παραμένουν στο σακουλάκι θα πρέπει να είναι γνωστή στον παίκτη.
- Το πρόγραμμα στη συνέχεια περιμένει τον παίκτη να πληκτρολογήσει μια λέξη με τα γράμματα που διαθέτει (ένα παράδειγμα οθόνης βλέπετε παρακάτω – είναι απολύτως ενδεικτικό – δεν εννοώ να εμφανίζει ακριβώς αυτά τα μηνύματα και η δική σας εφαρμογή)

```
>>>
Στο σακουλάκι: 75 γράμματα - Παίζεις:
Διαθέσιμα Γράμματα: Ε,1 - Β,8 - Α,1 - Χ,8 - Μ,3 - Η,1 - Θ,10
Λέξη:
```

-- Σενάριο Παιχνιδιού 3/7

- **3)** Όταν ο παίκτης πληκτρολογήσει μια λέξη τότε το πρόγραμμά σας πρέπει να κάνει τουλάχιστον τους εξής ελέγχους:
- 3-Α) Ελέγχει αν η λέξη αποτελείται από γράμματα που όντως διαθέτει ο παίκτης. Αν **όχι**, εμφανίζει ένα σχετικό μήνυμα και περιμένει νέα λέξη
- 3-Β) Αν **ναι**, τότε ελέγχει αν η λέξη που δόθηκε περιλαμβάνεται στον κατάλογο αποδεκτών λέξεων που έχει προκύψει από το αρχείο **greek7.txt**.
- 3-Γ) Αντί για λέξη ο παίκτης αν θέλει **μπορεί να πληκτρολογήσει 'p'** (δηλ. pass) οπότε το πρόγραμμα θα πρέπει: α) να κληρώσει νέα γράμματα για τον παίκτη και β) μετά να επιστρέψει τα προηγούμενα γράμματα του παίκτη στο «σακουλάκι».
- Στη συνέχεια περνά στο βήμα 5 που περιγράφεται σε επόμενη διαφάνεια.

- **Πού βρίσκεται το αρχείο greek7.txt;**
- Είναι διαθέσιμο στη σελίδα του μαθήματος. Το πώς κατασκευάζεται και τι περιλαμβάνει το εξηγούμε στο εργαστήριο

-- Σενάριο Παιχνιδιού 4/7

- **4)** Αν η λέξη που **πληκτρολόγησε** ο παίκτης **είναι αποδεκτή** το πρόγραμμα **υπολογίζει τους πόντους της λέξης** και εμφανίζει στον παίκτη **το νέο του σκορ**, μαζί με μια προτροπή της μορφής '*Enter για συνέχεια*' (πχ. δείτε το παράδειγμα των μηνυμάτων παρακάτω)

```
>>>
Στο σακουλάκι: 75 γράμματα - Παίζεις:
Διαθέσιμα Γράμματα: Ε,1 - Β,8 - Α,1 - Χ,8 - Μ,3 - Η,1 - Θ,10
Λέξη: ΜΕΘΗ
Αποδεκτή Λέξη - Βαθμοί: 15 - Σκορ: 55
Enter για Συνέχεια
```

-- Σενάριο Παιχνιδιού 5/7

- **5)** Εφόσον δοθεί ‘Enter’ (ή ‘p’) από τον παίκτη στη συνέχεια το πρόγραμμα:
- 5-1) Συμπληρώνει με νέα γράμματα τα διαθέσιμα του παίκτη (ώστε να είναι πάντοτε 7, θυμηθείτε ότι ταυτόχρονα πρέπει να τα αφαιρεί από το «σακουλάκι»).
- 5-2) Εμφανίζει τα γράμματα του υπολογιστή και την λέξη που παίζει, μαζί με τη βαθμολογία λέξης και το συνολικό σκορ του υπολογιστή. Συμπληρώνει κρυφά τα γράμματα του Η/Υ και τέλος επανέρχεται στο βήμα 3.
- Παράδειγμα οθόνης με τα βήματα αυτά βλέπετε παρακάτω.

```
Enter για Συνέχεια
--------------------------------------------------
Διαθέσιμα Γράμματα: Γ,4 - Β,8 - Α,1 - Χ,8 - Κ,2 - Α,1 - Ο,1
--------------------------------------------------
Στο σακουλάκι: 71 γράμματα - Παίζει ο Η/Υ:
Γράμματα Η/Υ: Σ,1 - Ν,1 - Ο,1 - Φ,8 - Ζ,10 - Η,1 - Ε,1
Λέξη Η/Υ: ΝΕΦΟΣ, Βαθμοί: 12 - Σκορ Η/Υ: 42
--------------------------------------------------
Στο σακουλάκι: 66 γράμματα - Παίζεις:
Διαθέσιμα Γράμματα: Γ,4 - Β,8 - Α,1 - Χ,8 - Κ,2 - Α,1 - Ο,1
Λέξη:
```

-- Σενάριο Παιχνιδιού 6/7

- **6)** Το παιχνίδι συνεχίζεται έτσι μέχρις ότου συμβεί κάτι από τα παρακάτω:
- 6-1) Ο παίκτης επιθυμεί να σταματήσει (**ή** δεν βρίσκει αποδεκτή λέξη να παίξει **και** δεν υπάρχουν γράμματα για να αλλάξει) οπότε εισάγει τον χαρακτήρα **‘q’** όταν είναι η σειρά του να πληκτρολογήσει λέξη.
- 6-2) Δεν υπάρχουν πλέον γράμματα αρκετά στο «σακουλάκι» ώστε να αντικατασταθούν όσα λείπουν, είτε του παίκτη είτε του Η/Υ
- 6-3) Ο Η/Υ δεν βρίσκει κάποια αποδεκτή λέξη να παίξει

-- Σενάριο Παιχνιδιού 7/7

- **7)** Σε οποιαδήποτε από τις παραπάνω περιπτώσεις το πρόγραμμα σταματά και:
- 7-1) Ανακοινώνει τις βαθμολογίες Παίκτη και Η/Υ ανακηρύσσοντας τον νικητή
- 7-2) Καταχωρεί σε **κατάλληλη δομή δεδομένων που αποθηκεύεται σε αρχείο** μια νέα καταχώρηση με όποια στοιχεία του παιχνιδιού κρίνονται σημαντικά (πχ. πόσες κινήσεις παίχτηκαν, ποιο ήταν το σκορ παίκτη και Η/ Υ). Την δομή αυτή το πρόγραμμα την «φορτώνει» όταν ξεκινά το παιχνίδι και μπορεί να δίνει στον παίκτη σχετική ενημέρωση.
- Για να αποθηκεύσετε δεδομένα σε αρχείο θα χρησιμοποιήσετε την **βιβλιοθήκη pickle ή json (προτιμήστε json)** για την οποία μπορείτε να διαβάσετε στο πακέτο διαφανειών 09-Python-Files.pdf στην ενότητα **'Διατήρηση' (pickle & json)**

(4) Δομή δεδομένων για τις αποδεκτές λέξεις

Δομή δεδομένων για τις λέξεις: σε Λεξικό (dictionary) ή σε Λίστα (list);

- Το πρόγραμμά σας στο ξεκίνημα θα πρέπει να κάνει μια ακόμη σημαντική εργασία:
- Να φορτώνει τις λέξεις της γλώσσας από το αρχείο **greek7.txt** και να τις μεταφέρει σε μια κατάλληλη δομή στον κώδικα ώστε να μπορούν να γίνουν οι απαραίτητοι έλεγχοι στο παιχνίδι
- Θα πρέπει να αποφασίσετε για τη **δομή δεδομένων** που θα περιέχει τις λέξεις της γλώσσας που θα συμβουλεύεται το πρόγραμμά σας.
- Από τη είδος της δομής θα εξαρτηθεί η **αποδοτικότητα του κώδικά** σας. Το συζητήσαμε αυτό στο εργαστήριο και έχουμε δώσει την απάντηση.

(5) Αλγόριθμος play

Πώς θα παίζει ο Υπολογιστής; ΣΕΝΑΡΙΟ 1

- Ο αλγόριθμος παιχνιδιού του Η/Υ είναι ο **MIN-MAX-SMART**
- Δηλ. μπορεί να έχει κατ' επιλογή του παίκτη μία από τις **τρεις** παρακάτω μορφές:
- Α) **MIN Letters:** Το πρόγραμμα δημιουργεί όλες τις δυνατές **μεταθέσεις** (**permutations**) των γραμμάτων που διαθέτει ο Η/Υ ξεκινώντας από 2 και ανεβαίνοντας μέχρι τα 7 γράμματα. Για κάθε μετάθεση ελέγχει αν είναι αποδεκτή λέξη και παίζει την πρώτη αποδεκτή λέξη που θα εντοπίσει.
- Β) **MAX Letters:** Όπως και στο Α αλλά το πρόγραμμα ξεκινά από τις μεταθέσεις των γραμμάτων ανά 7 και **κατεβαίνει** προς το 2. Παίζει πάλι την πρώτη αποδεκτή λέξη αλλά τώρα αυτή με τα περισσότερα γράμματα.
- Γ) **SMART:** Όπως και στο Α αλλά **εξαντλεί** όλες τις μεταθέσεις 2 ως και 7 γραμμάτων χωρίς να σταματά. Βρίσκει τις αποδεκτές λέξεις και στο τέλος παίζει τη λέξη που δίνει τους περισσότερους βαθμούς.

- Οι μεταθέσεις μιας λίστας αντικειμένων μπορούν εύκολα να είναι διαθέσιμες στον κώδικά σας μέσω της συνάρτησης itertools.permutations()

Πώς θα παίζει ο Υπολογιστής; ΣΕΝΑΡΙΟ 2

- Ο αλγόριθμος παιχνιδιού του Η/Υ έχει την μορφή: **SMART-FAIL**

- **SMART**: Καλείται πρώτα ο αλγόριθμος SMART (όπως εξηγήθηκε στο σενάριο 1) και δημιουργεί μια λίστα με πιθανές λέξεις που μπορούν να παιχτούν
- **FAIL**: Στη συνέχεια καλείται ο FAIL. Όπως ένα άνθρωπος δεν βρίσκει πάντοτε τη βέλτιστη λέξη ο αλγόριθμος FAIL εισάγει ένα βαθμό αποτυχίας στο να παίξει τη βέλτιστη λέξη που έχει βρει ο SMART.
- Ο αλγόριθμος FAIL παίρνει ως είσοδο τη λίστα πιθανών λέξεων που παράγει ο SMART και επιλέγει να παίξει πχ. τη $2^{η}$ καλύτερη ή την $3^{η}$ καλύτερη λέξη στη λίστα (αντί της βέλτιστης πρώτης) ανάλογα πώς θα τον γράψετε.
- Μπορείτε να **αποφασίσετε εσείς τις λεπτομέρειες του αλγορίθμου FAIL** έτσι ώστε ο FAIL να προσομοιώνει τη μνήμη/ικανότητα ενός ανθρώπου που δεν γνωρίζει όλες τις λέξεις ή δεν μπορεί να βρει πάντοτε την καλύτερη δυνατή λέξη.

Πώς θα παίζει ο Υπολογιστής; ΣΕΝΑΡΙΟ 3

- Ο αλγόριθμος παιχνιδιού του Η/Υ έχει την μορφή: **SMART-EXPERT**

- **SMART**: Ο αλγόριθμος όπως εξηγήθηκε στο σενάριο 1
- **EXPERT**: προσομοιώνει έναν «ειδικό» σε κάποιο πεδίο ο οποίος μπορεί να γνωρίζει ειδικούς όρους και επομένως περισσότερες λέξεις από αυτές που περιέχει το αρχείο greek7.txt. Πχ. το όνομα ‘Σφίγξ’ δεν υπάρχει στο greek7.txt αλλά θα μπορούσε να περιλαμβάνεται σε ένα λεξικό με αρχαϊκούς τύπους λέξεων
- Στο σενάριο αυτό θα πρέπει να εμπλουτίσετε (ή αντικαταστήσετε) το αρχείο greek7.txt με άλλο το οποίο να περιέχει μεγαλύτερο ή εξειδικευμένο πλούτο λέξεων (θυμηθείτε: μέχρι 7 γράμματα).
- Δηλ. ουσιαστικά πρόκειται για τον αλγόριθμο SMART αλλά ο οποίος εφαρμόζεται σε εμπλουτισμένο αρχείο λέξεων.

Πώς θα παίζει ο Υπολογιστής; ΣΕΝΑΡΙΟ 4

- Ο αλγόριθμος παιχνιδιού του Η/Υ έχει την μορφή: **SMART-LEARN**

- **SMART**: Ο αλγόριθμος όπως εξηγήθηκε στο σενάριο 1
- **LEARN**: Ο υπολογιστής «μαθαίνει» νέες λέξεις, δηλ. ο αλγόριθμος προσομοιώνει έναν παίκτη ο οποίος μαθαίνει καθώς παίζει.
- Δηλ. αν ο παίκτης-άνθρωπος σε κάποια κίνηση εισάγει μια **λέξη που δεν υπάρχει** στο αρχείο greek7.txt (αλλά είναι μέχρι 7 γράμματα) τότε αντί να απορριφθεί ερωτάται ο παίκτης αν θέλει **να συμπεριληφθεί στο αρχείο λέξεων** για τη συνέχεια.
- Εφόσον ο παίκτης επιλέξει 'ΝΑΙ' η λέξη συμπεριλαμβάνεται στη δομή λέξεων του τρέχοντος παιχνιδιού ώστε να αναγνωρίζεται στη συνέχεια.
- Στο σενάριο αυτό θα πρέπει στο τέλος του παιχνιδιού να ενημερώσετε το αρχείο greek7.txt με τις νέες λέξεις ώστε να είναι διαθέσιμες σε επόμενη παρτίδα.
- Μπορείτε να σχεδιάσετε εσείς τις τεχνικές-προγραμματιστικές λεπτομέρειες του σεναρίου

Πώς θα παίζει ο Υπολογιστής; ΣΕΝΑΡΙΟ 5

- Ο αλγόριθμος παιχνιδιού του Η/Υ έχει την μορφή: **SMART-TEACH**

- **SMART**: Ο αλγόριθμος όπως εξηγήθηκε στο σενάριο 1
- **TEACH**: Ο υπολογιστής «διδάσκει» τον παίκτη, δηλ. τον ενημερώνει ποια θα ήταν η καλύτερη λέξη να παίξει.
- Όταν είναι σειρά του παίκτη-ανθρώπου να παίξει, ο υπολογιστής εκτελεί επίσης τον αλγόριθμο SMART.
- Αφού παίξει ο παίκτη κάποια λέξη ο υπολογιστής ενημερώνει τον παίκτη αν η λέξη που έπαιξε είναι η καλύτερη δυνατή. Αν δεν είναι, τότε τον ενημερώνει ποια θα ήταν η καλύτερη ή και η $2^{η}$ καλύτερη λέξη που θα μπορούσε να παίξει.
- Μπορείτε να σχεδιάσετε εσείς τις τεχνικές-προγραμματιστικές λεπτομέρειες του σεναρίου.

ΣΕΝΑΡΙΑ και ΡΥΘΜΙΣΕΙΣ

- Στην εργασία σας θα πρέπει **να επιλέξετε ποιο σενάριο από τα προηγούμενα θα υλοποιήσετε στον κώδικά σας.**
- Αν η εργασία γίνει από **ομάδα 2 ατόμων** θα πρέπει να υλοποιηθούν **2 διαφορετικά σενάρια**.

- Όταν ο παίκτης επιλέξει '**Ρυθμίσεις**' στην αρχή του παιχνιδιού το πρόγραμμά σας να παρουσιάζει τα δυνατά σενάρια και να επιτρέπει στον παίκτη να διαλέξει.

- Αν σκέφτεστε άλλα ενδιαφέροντα σενάρια μπορείτε να τα υλοποιήσετε αφού πρώτα **με ενημερώσετε και πάρετε την έγκρισή μου**

(6) Βιβλιοθήκη & import

Module (Βιβλιοθήκη, άρθρωμα)

- Ένα ‘module’ (‘**βιβλιοθήκη**’ ή και ‘άρθρωμα’) είναι ένα αρχείο .py στο οποίο έχετε αποθηκεύσει κώδικα που κάνει συγκεκριμένες εργασίες, πχ. κλάσεις ή και απλές συναρτήσεις

- Μια τέτοια βιβλιοθήκη συνδέεται με το κύριο πρόγραμμά σας με εντολή **import**
- Πχ. ας υποθέσουμε πως έχετε γράψει την κλάση SakClass και την έχετε αποθηκεύσει σε ένα αρχείο classes.py
- Τότε στο κύριο πρόγραμμά σας μπορείτε να γράψετε:
- **import classes**
- Και να δημιουργήσετε το αντικείμενο sak τύπου SakClass γράφοντας:
- **sak = classes.SakClass()**

(7) Τεκμηρίωση & docstring

(7) Τεκμηρίωση (με μορφή docstring)

- Συμπεριλάβετε τεκμηρίωση του κώδικά σας σε μορφή **docstring** στην αρχή του κύριου προγράμματος ενσωματωμένο σε μια συνάρτηση με όνομα '**guidelines**'
- Δείτε την επόμενη ενότητα ('Αξιολόγηση' για να ξέρετε το τι θα συμπεριλάβετε στην τεκμηρίωση
- Επίσης συμπεριλάβετε κάθε χρήσιμη πληροφορία για τον τρόπο που εκτελείτε το πρόγραμμά σας και που πρέπει να ξέρει κάποιος που θέλει να παίξει το παιχνίδι σας.

- Το docstring τεκμηρίωσης θα πρέπει να εμφανίζεται όταν πληκτρολογηθεί: >>> **help(guidelines)**
- **Δοκιμάστε ότι το παραπάνω δουλεύει σωστά πριν υποβάλετε την εργασία σας.**

(8) Αξιολόγηση εργασίας

(8) Αξιολόγηση 1/2

- Η αξιολόγηση της εργασίας θα γίνει με βάση τα παρακάτω

ΚΡΙΤΗ-ΡΙΟ	ΧΑΡΑΚΤΗΡΙ-ΣΤΙΚΟ	ΣΧΟΛΙΑ - ΕΞΗΓΗΣΕΙΣ
Α	**ΚΛΑΣΕΙΣ**	Αν έχουν υλοποιηθεί οι κλάσεις όπως έχουν περιγραφεί. Ποιες μεθόδους έχετε υλοποιήσει σε κάθε κλάση και τι κάνουν (2 γραμμές για την καθεμιά). Επίσης αν έχετε υλοποιήσει άλλες κλάσεις *- (εξηγήστε όλα τα παραπάνω στο docstring)*
Β	**ΛΙΣΤΑ, ΛΕΞΙΚΟ για τη Δομή των Λέξεων**	Εξηγήστε ποια είναι η δομή που χρησιμοποιήσατε *- (εξηγήστε το στο docstring)*
Γ	**ΒΙΒΛΙΟΘΗΚΗ**	Οι κλάσεις να βρίσκονται σε ιδιαίτερη βιβλιοθήκη (ξεχωριστό αρχείο) όπως έχει περιγραφεί.
Δ	**ΑΛΓΟΡΙΘΜΟΣ play**	Ποιο ή ποια σενάρια για τον αλγόριθμο play έχουν υλοποιηθεί *- (εξηγήστε το στο docstring ειδικά αν έχετε υλοποιήσει κάποιο ιδιαίτερο σενάριο που δεν προτάθηκε εδώ)*

- ***Συνέχεια κριτηρίων αξιολόγησης …*** →

(8) Αξιολόγηση 2/2

ΚΡΙΤΗ-ΡΙΟ	ΧΑΡΑΚΤΗΡΙ-ΣΤΙΚΟ	ΣΧΟΛΙΑ - ΕΞΗΓΗΣΕΙΣ
Ε	**ΟΡΘΟΤΗΤΑ ΚΩΔΙΚΑ - ΜΗΝΥΜΑΤΑ**	Αξιολογείται γενικά η **ορθότητα** του κώδικα ώστε να παίζεται σωστά το παιχνίδι καθώς και η κατανοητή **εμφάνιση** των μηνυμάτων και **υλοποίηση** των προβλεπόμενων λειτουργιών. *Μπορείτε να σχεδιάσετε την εμφάνιση των μηνυμάτων όπως θέλετε και όχι οπωσδήποτε σύμφωνα με τα προηγούμενα παραδείγματα που δίνω εδώ.*
ΣΤ	**ΤΕΚΜΗΡΙΩΣΗ**	Να περιλαμβάνεται στο docstring κάθε πληροφορία όπως ζητώ στον πίνακα αυτό ή οτιδήποτε άλλο είναι σημαντικό για την κατανόηση της λειτουργίας του κώδικά σας **ΠΡΟΣΟΧΗ: Έλλειψη τεκμηρίωσης στο docstring μειώνει τη βαθμολογία σας.**
Ζ	**ΑΡΧΕΙΑ**	Αν έχουν υποβληθεί **όλα τα αρχεία** που είναι απαραίτητα για να εκτελείται σωστά ο κώδικάς σας. *- (εξηγήστε το στο docstring από ποια αρχεία αποτελείται η εργασία σας)* **ΠΡΟΣΟΧΗ: Λάθος εκτέλεση του κώδικα λόγω έλλειψης κάποιου αρχείου μειώνει τη βαθμολογία σας.**

(9) Παραδοτέο

(9) Παραδοτέο

- Παραδοτέο της εργασίας σας είναι **1 zip ή rar αρχείο που περιλαμβάνει:**

- (α) Το αρχείο **classes.py** που περιλαμβάνει τις **κλάσεις** της εργασίας σας
- (β) Το αρχείο **main-AEM.py** που περιλαμβάνει το **κύριο πρόγραμμα**
  - Πχ. main1234.py ή main12343005.py σε περίπτωση 2 συνεργατών με ΑΕΜ 1234 και 3005 (τα ΑΕΜ συνεχόμενα)
- (γ) Κάθε άλλο **απαραίτητο αρχείο** όπως εσείς θεωρείτε σωστό για την εκτέλεση του κώδικα

- >>> Τα παραπάνω αρχεία τα **συμπιέζετε σε αρχείο zip ή rar**
- >>> Ονομάζετε το συμπιεσμένο αρχείο: **PythonScrabbleAEM** (ή βάζετε AEM1AEM2 αν είστε 2 συνεργάτες)
- **Υποβάλετε μέσω elearning** μέχρι την **προθεσμία** που θα ανακοινωθεί στην εξεταστική που θέλετε να παραδώσετε την εργασία

Υπενθύμιση για τις εξετάσεις του μαθήματος

- Το μάθημα έχει **3 εξετάσεις**:
- **Θ1**: Τεστ Python (εξέταση στο τέλος του εργαστηρίου) 20%
- **Θ2**: Εργασία (Project) Python Scrabble 40%
- **Θ3**: Γραπτά 40%

- Μπορείτε να "**μοιράσετε**" τις εξετάσεις όπως θέλετε στις εξεταστικές: Ιουνίου & Σεπτεμβρίου 2019 (και Φεβρουαρίου 2020 αν είστε επί πτυχίω)

- *Προσοχή*: Εφόσον χρωστάτε το μάθημα και ξαναδιδαχθεί το εαρινό 2020 υποχρεούστε σε συνολική επανεξέταση **(δεν κρατούνται βαθμοί μετά τον Φεβρουάριο 2020)**