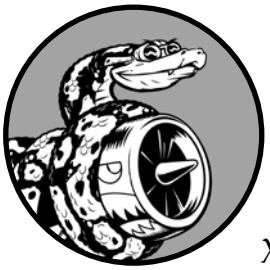
2

ΜΕΤΑΒΛΗΤΕΣ ΚΑΙ ΑΠΛΟΙ ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ



Στο κεφάλαιο αυτό θα μάθουμε διάφορα είδη δεδομένων που μπορούν να δουλέψουν στα προγράμματα Python. Επίσης, θα μάθουμε να χρησιμοποιούμε μεταβλητές προκειμένου να αναπαραστήσουμε δεδομένα στα προγράμματά μας.

Τι συμβαίνει πραγματικά όταν εκτελούμε το hello_world.py

Ας δούμε καλύτερα τι πραγματικά κάνει η Python όταν εκτελούμε το hello_world.py. Όπως φαίνεται, η Python κάνει αρκετή δουλειά, ακόμη κι όταν τρέχουμε ένα απλό πρόγραμμα:

hello_world.py

```
print("Hello Python world!")
```

Όταν εκτελούμε τον συγκεκριμένο κώδικα, θα πρέπει να βλέπουμε το συγκεκριμένο αποτέλεσμα:

```
Hello Python world!
```

Όταν εκτελούμε το αρχείο hello_world.py, η κατάληξη .py υποδηλώνει ότι το αρχείο είναι ένα πρόγραμμα στη γλώσσα Python. Έπειτα, ο επεξεργαστής (editor) εκτελεί το αρχείο με τον Python interpreter, τον διερμηνέα της Python, ο οποίος διατρέχει το πρόγραμμα και προσδιορίζει τι σημαίνει κάθε λέξη που περιέχει το πρόγραμμα. Για παράδειγμα, όταν ο διερμηνέας δει τη λέξη print ακολουθούμενη από παρένθεση, εμφανίζει στην οθόνη οτιδήποτε βρίσκεται μέσα στην παρένθεση.

Καθώς γράφουμε τα προγράμματα, ο επεξεργαστής τονίζει διά-

φορα μέρη του προγράμματος με διαφορετικούς τρόπους. Για παράδειγμα, αναγνωρίζει ότι το print() είναι το όνομα μιας λειτουργίας και εμφανίζει τη συγκεκριμένη λέξη με ένα χρώμα. Αναγνωρίζει ότι το «Hello Python world!» δεν είναι κώδικας της Python και εμφανίζει τη συγκεκριμένη φράση με ένα διαφορετικό χρώμα. Η ιδιότητα αυτή ονομάζεται syntax highlighting (χρωματισμός της σύνταξης) και είναι αρκετά χρήσιμη όταν ξεκινάμε να γράφουμε δικά μας προγράμματα.

Μεταβλητές

Ας προσπαθήσουμε να χρησιμοποιήσουμε μία μεταβλητή (variable) στο hello_world.py. Προσθέτουμε μία νέα γραμμή στην αρχή του αρχείου και τροποποιούμε τη δεύτερη γραμμή:

hello_world.py

```
message = "Hello Python world!"
print(message)
```

Τρέξτε το πρόγραμμα και δείτε τι συμβαίνει. Θα πρέπει να δείτε το ίδιο αποτέλεσμα που είδατε και νωρίτερα:

```
Hello Python world!
```

Προσθέσαμε μία μεταβλητή που ονομάζεται message. Κάθε μεταβλητή συνδέεται με μία τιμή (value), η οποία αναπαριστά την πληροφορία που σχετίζεται με τη συγκεκριμένη μεταβλητή. Στη συγκεκριμένη περίπτωση, η τιμή είναι το κείμενο «Hello Python world!».

Η προσθήκη της μεταβλητής κάνει λίγο δυσκολότερη τη δουλειά του διερμηνέα της Python. Όταν επεξεργάζεται την πρώτη γραμμή, συνδέει τη μεταβλητή message με το κείμενο «Hello Python world!». Όταν φτάνει στη δεύτερη γραμμή, εμφανίζει στην οθόνη την τιμή που σχετίζεται με το message.

Ας διευρύνουμε το συγκεκριμένο πρόγραμμα τροποποιώντας το hello_world.py έτσι ώστε να εμφανίσει κι ένα δεύτερο μήνυμα. Προσθέτουμε μία κενή γραμμή στο hello_world.py, κι έπειτα προσθέτουμε δύο νέες γραμμές κώδικα:

```
message = "Hello Python world!"
print(message)

message = "Hello Python Crash Course world!"
print(message)
```

Τώρα όταν εκτελέσουμε το hello_world.py, θα δούμε δύο γραμμές μηνύματος:

```
Hello Python world!
Hello Python Crash Course world!
```

Μπορούμε να αλλάξουμε την τιμή μιας μεταβλητής στο πρόγραμμά μας οποιαδήποτε στιγμή και η Python θα ακολουθεί πάντα την τρέχουσα τιμή.

Ονομασία και χρήση μεταβλητών

Όταν χρησιμοποιούμε μεταβλητές στην Python, υπάρχουν κάποιοι κανόνες και οδηγίες που πρέπει να τηρούμε. Η παραβίαση κάποιου από τους κανόνες θα προκαλέσει σφάλματα. Επιπλέον, κάποιες οδηγίες μας βοηθούν να γράψουμε κώδικα με τρόπο που θα είναι πιο εύκολο να τον διαβάσουμε και να τον κατανοήσουμε. Ας δούμε κάποιους κανόνες μεταβλητών που πρέπει να έχουμε υπόψη:

Τα ονόματα των μεταβλητών μπορούν να περιέχουν μόνο γράμματα, αριθμούς και κάτω παύλες. Μπορούν να ξεκινούν με ένα γράμμα ή μία κάτω παύλα, αλλά όχι με αριθμό. Για παράδειγμα, μπορούμε να έχουμε μία μεταβλητή message_1 αλλά όχι 1_message.

Δεν επιτρέπονται κενά στα ονόματα των μεταβλητών, παρά μόνο κάτω παύλες για τον διαχωρισμό των ονομάτων. Για παράδειγμα, το όνομα greeting_message θα είναι αποδεκτό, αλλά το greeting message θα προκαλέσει σφάλματα.

Αποφεύγουμε να χρησιμοποιούμε λέξεις κλειδιά της Python και ονόματα συναρτήσεων ως ονόματα μεταβλητών. Με άλλα λόγια, δεν θα πρέπει να χρησιμοποιούμε λέξεις που υπάρχουν για συγκεκριμένο σκοπό προγραμματισμού όπως, για παράδειγμα, η λέξη print. (Βλ. «Λέξεις Κλειδιά της Python και Ενσωματωμένες Συναρτήσεις» στη σελίδα 568).

Τα ονόματα των μεταβλητών θα πρέπει να είναι μικρά σε έκταση αλλά περιγραφικά. Για παράδειγμα, το name είναι καλύτερο από το n, το student_name είναι καλύτερο από το s_n, και το name_length είναι καλύτερο από το length_of_persons_name.

Θα πρέπει να προσέχουμε όταν χρησιμοποιούμε το πεζό γράμμα l και το κεφαλαίο O, καθώς μπορεί να υπάρξει σύγχυση με τους αριθμούς 1 και 0.

Χρειάζεται εξάσκηση για να μάθουμε να δημιουργούμε καλά ονόματα μεταβλητών, ιδίως όσο τα προγράμματα θα γίνονται πιο ενδιαφέροντα και πιο περίπλοκα. Όσο θα γράφετε περισσότερα προγράμματα και θα αρχίσετε να διαβάζετε μέσα από κώδικα άλλων, θα βελτιώνεστε και στην ονοματοδοσία των μεταβλητών.

ΣΗΜΕΙΩΣΗ

Οι μεταβλητές της Python που θα χρησιμοποιήσουμε σε αυτό το στάδιο θα είναι σε πεζά γράμματα. Δεν θα δημιουργηθούν σφάλματα αν χρησιμοποιήσετε κεφαλαία γράμματα, απλά τα κεφαλαία γράμματα σε ονόματα μεταβλητών έχουν συγκεκριμένες σημασίες που θα συζητηθούν σε επόμενα κεφάλαια.

5

ΕΝΤΟΛΕΣ IF

Ο προγραμματιστής συχνά καλείται να εξετάσει ένα σύνολο συνθηκών και να αποφασίσει σε ποια ενέργεια να προχωρήσει με βάση τις συγκεκριμένες συνθήκες. Η εντολή if της Python μας επιτρέπει να εξετάσουμε την τρέχουσα κατάσταση ενός προγράμματος και να ανταποκριθούμε κατάλληλα. Σε αυτό το κεφάλαιο θα μάθουμε να γράφουμε ελέγχους υπό συνθήκη, οι οποίοι μας επιτρέπουν να ελέγχουμε οποιαδήποτε κατάσταση μας ενδιαφέρει. Θα μάθουμε να γράφουμε απλές εντολές if, δηλαδή if statements ή συνθήκες if, και να δημιουργούμε πιο περίπλοκες σειρές από εντολές if για να εντοπίζουμε πότε παρίστανται οι ακριβείς συνθήκες που θέλουμε. Έπειτα, θα εφαρμόσουμε αυτές τις έννοιες σε λίστες, ώστε να μπορούμε να γράψουμε βρόχο for που να χειρίζεται τα περισσότερα στοιχεία μιας λίστας αλλά και μεμονωμένες τιμές με διαφορετικό τρόπο.

Ένα απλό παράδειγμα

Το παρακάτω σύντομο παράδειγμα μας δείχνει πώς οι έλεγχοι if μας επιτρέπουν να απαντάμε σωστά σε ειδικές περιστάσεις. Φανταστείτε ότι έχετε μια λίστα αυτοκινήτων και θέλετε να εμφανίσετε το όνομα κάθε αυτοκινήτου. Τα ονόματα των αυτοκινήτων είναι τα επίσημα ονόματα, οπότε θα πρέπει να εμφανισθούν με γράμματα τίτλου. Ωστό-

σο, η τιμή 'bmw' θα πρέπει να εμφανιστεί με όλα κεφαλαία γράμματα. Ο παρακάτω κώδικας διατρέχει με βρόχο μια λίστα με ονόματα αυτοκινήτων και αναζητά την τιμή 'bmw'. Όπου η τιμή είναι 'bmw', εμφανίζεται με κεφαλαία γράμματα αντί για γράμματα τίτλου:

cars.py

```
cars = ['audi', 'bmw', 'subaru', 'toyota']

for car in cars:
    if car == 'bmw':
        print(car.upper())
    else:
        print(car.title())
```

❶

Ο βρόχος σε αυτό το παράδειγμα ελέγχει πρώτα αν η τρέχουσα τιμή του car είναι 'bmw' ❶. Εάν ναι, τότε η τιμή εμφανίζεται με κεφαλαία γράμματα. Αν η τιμή car είναι οτιδήποτε άλλο εκτός 'bmw', τότε εμφανίζεται με γράμματα τίτλου:

```
Audi
BMW
Subaru
Toyota
```

Το συγκεκριμένο παράδειγμα συνδυάζει αρκετές έννοιες που θα μάθουμε σε αυτό το κεφάλαιο. Ας ξεκινήσουμε μελετώντας τα διάφορα είδη ελέγχων που μπορούμε να χρησιμοποιήσουμε για να εξετάσουμε τις συνθήκες στο πρόγραμμά μας.

Έλεγχοι υπό συνθήκη

Κάθε εντολή if εστιάζει σε μια έκφραση που μπορεί να αξιολογηθεί ως True ή False, δηλαδή σωστή ή λάθος, και ονομάζεται έλεγχος υπό συνθήκη ή αλλιώς conditional test. Η Python χρησιμοποιεί τις τιμές True και False για να αποφασίσει αν ο κώδικας σε μια εντολή if θα πρέπει να εκτελεστεί. Αν ο υπό συνθήκη έλεγχος αξιολογείται ως True, η Python εκτελεί τον κώδικα που ακολουθεί την εντολή if. Αν ο έλεγχος αξιολογείται ως False, η Python αγνοεί τον κώδικα που ακολουθεί την εντολή if.

Έλεγχος ισότητας

Τα περισσότερα conditional tests συγκρίνουν την τρέχουσα τιμή μιας μεταβλητής με μια συγκεκριμένη τιμή ενδιαφέροντος. Το πιο απλό conditional test ελέγχει αν η τιμή μιας μεταβλητής είναι ίση με την τιμή ενδιαφέροντος:

```
❶ >>> car = 'bmw'
❷ >>> car == 'bmw'
  True
```

Η γραμμή στο ❶ ορίζει την τιμή car σε 'bmw' χρησιμοποιώντας ένα μονό σύμβολο ισότητας, όπως έχουμε ήδη δει αρκετές φορές. Η γραμμή στο ❷ ελέγχει αν η τιμή car είναι 'bmw' χρησιμοποιώντας διπλό σύμβολο ισότητας (==). Ο τελεστής ισότητας (equality operator) επιστρέφει True αν οι τιμές στη δεξιά και αριστερή πλευρά του τελεστή ταιριάζουν και False αν δεν ταιριάζουν. Οι τιμές στο συγκεκριμένο παράδειγμα ταιριάζουν και έτσι η Python επιστρέφει True.

Αν η τιμή car είναι οτιδήποτε άλλο εκτός 'bmw', το test θα επιστρέφει False:

```
❶ >>> car = 'audi'
❷ >>> car == 'bmw'
  False
```

Ένα μονό σύμβολο ισότητας είναι στην πραγματικότητα μία δήλωση (statement). Μπορούμε να διαβάσουμε τον κώδικα στο ❶ ως "Ορίστε την τιμή της μεταβλητής car ίση με τη λέξη 'audi'." Από την άλλη μεριά, ένα διπλό σύμβολο ισότητας, όπως αυτό στο ❷, θέτει την ερώτηση: "Είναι η τιμή της μεταβλητής car ίση με τη λέξη 'bmw';" Οι περισσότερες γλώσσες προγραμματισμού χρησιμοποιούν τα σύμβολα ισότητας με βάση αυτό το σκεπτικό.

Παράλειψη πεζών/κεφαλαίων γραμμάτων κατά τον έλεγχο ισότητας

Στην Python, ο έλεγχος ισότητας είναι ένα θέμα ευαίσθητο όσον αφορά τη χρήση πεζών/κεφαλαίων γραμμάτων. Για παράδειγμα, δύο τιμές με διαφορετικό αρχικό γράμμα δεν θεωρούνται ίσες:

```
>>> car = 'Audi'
>>> car == 'audi'
False
```

Αν το γράμμα έχει σημασία, η συμπεριφορά αυτή είναι εξαιρετικά ωφέλιμη. Αν, όμως, το γράμμα δεν έχει σημασία και απλά θέλετε να ελέγξετε την τιμή μιας μεταβλητής, μπορείτε να μετατρέψετε την τιμή της μεταβλητής σε πεζά γράμματα προτού προβείτε στη σύγκριση:

```
>>> car = 'Audi'
>>> car.lower() == 'audi'
True
```

Το test αυτό θα επέστρεφε True ανεξάρτητα από τη μορφοποίηση της τιμής 'Audi' αφού πλέον το test δεν έχει ευαισθησία σε πεζά/κεφαλαία γράμματα. Η συνάρτηση lower() δεν αλλάζει την τιμή που είχε αρχικά αποθηκευτεί στο στοιχείο car, επομένως μπορείτε να κάνετε αυτού του είδους τη σύγκριση χωρίς να επηρεάσετε την αρχική μεταβλητή:

```
❶ >>> car = 'Audi'
❷ >>> car.lower() == 'audi'
```

2ο Μέρος

ΠΡΟΤΖΕΚΤ

Συγχαρητήρια! Γνωρίζετε ήδη αρκετά για την Python για να ξεκινήσετε να δημιουργείτε και να αναπτύσσετε διαδραστικά προγράμματα με νόημα και σκοπό. Η διαδικασία δημιουργίας των δικών σας πρότζεκτ θα σας διδάξει νέες δεξιότητες και θα ενισχύσει την κατανόηση των εννοιών που παρουσιάστηκαν στο 1ο Μέρος.

Το 2ο Μέρος περιέχει τρεις τύπους πρότζεκτ και μπορείτε να επιλέξετε να ασχοληθείτε με οποιοδήποτε ή και με όλα τα πρότζεκτ με όποια σειρά θέλετε. Ακολουθεί μια σύντομη περιγραφή του κάθε πρότζεκτ, η οποία θα σας βοηθήσει να αποφασίσετε σε ποιο να εντρυφήσετε πρώτα.

Alien Invasion: Δημιουργήστε ένα παιχνίδι με Python

Στο πρότζεκτ Alien Invasion (Κεφάλαια 12, 13 και 14), θα χρησιμοποιήσουμε το πακέτο Pygame για να δημιουργήσουμε ένα παιχνίδι 2D, με επίπεδα αυξανόμενης ταχύτητας και δυσκολίας, όπου στόχος θα είναι να αποτρέψουμε έναν στόλο εξωγήινων που θα εμφανίζονται στην οθόνη. Με την ολοκλήρωση αυτού του πρότζεκτ, θα έχετε μάθει δεξιότητες με τις οποίες θα μπορέσετε να δημιουργήσετε δικά σας παιχνίδια 2D στο Pygame.

Οπτικοποίηση δεδομένων

Το πρότζεκτ της Οπτικοποίησης Δεδομένων ξεκινά στο Κεφάλαιο 15, όπου θα μάθουμε να παράγουμε δεδομένα και να δημιουργούμε μια σειρά λειτουργικές και ευπαρουσίαστες οπτικοποιήσεις δεδομένων χρησιμοποιώντας Matplotlib και Plotly. Στο Κεφάλαιο 16 θα μάθουμε πώς να προσπελάσουμε δεδομένα από διαδικτυακές πηγές και να τα εφοδιάσουμε σε ένα πακέτο οπτικοποίησης για να δημι-

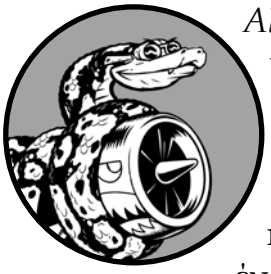
ουργήσουμε γραφικές παραστάσεις δεδομένων για τον καιρό και έναν παγκόσμιο χάρτη για την σεισμική δραστηριότητα. Τέλος, στο Κεφάλαιο 17 θα ξεκινήσουμε να γράφουμε πρόγραμμα για να κατεβάζουμε και να οπτικοποιούμε δεδομένα αυτόματα. Η εκμάθηση δημιουργίας οπτικοποιήσεων θα σας επιτρέψει να εισέλθετε στον χώρο της εξόρυξης δεδομένων (data mining), ένα από τα σημαντικότερα προσόντα και δεξιότητες των σημερινών προγραμματιστών.

Διαδικτυακές εφαρμογές

Στο πρότζεκτ των Διαδικτυακών Εφαρμογών (Κεφάλαια 18, 19 και 20), θα χρησιμοποιήσουμε το πακέτο Django για να δημιουργήσουμε μια απλή διαδικτυακή εφαρμογή που θα επιτρέπει στους χρήστες να διατηρούν ημερολόγιο για πληροφορίες για οποιαδήποτε θέματα μαθαίνουν. Οι χρήστες θα χρησιμοποιούν έναν λογαριασμό με username και password, θα εισάγουν ένα θέμα και έπειτα θα κάνουν επιμέρους καταχωρήσεις για το τι μαθαίνουν σχετικά με αυτό. Θα μάθουμε, επίσης, να αναπτύσσουμε την εφαρμογή ώστε να μπορεί ο καθένας να έχει πρόσβαση από όπου κι αν βρίσκεται. Με την ολοκλήρωση αυτού του πρότζεκτ, θα είστε σε θέση να δημιουργήσετε τις δικές σας απλές διαδικτυακές εφαρμογές και θα είστε έτοιμοι να εμβαθύνετε σε πιο λεπτομερείς πηγές για τη δημιουργία εφαρμογών με Django.

14

ΒΑΘΜΟΛΟΓΙΑ



Σε αυτό το κεφάλαιο, θα ολοκληρώσουμε το παιχνίδι *Alien Invasion*. Θα προσθέσουμε το κουμπί Play για να μπορεί ο παίκτης να ξεκινά το παιχνίδι κατ' επιλογή ή και να επανεκκινεί ένα παιχνίδι μόλις αυτό τερματιστεί. Θα αλλάξουμε το παιχνίδι ώστε να αυξάνει σε ταχύτητα όταν ο παίκτης περνά σε επόμενο επίπεδο και θα υλοποιήσουμε ένα σύστημα βαθμολόγησης (scoring system). Με την ολοκλήρωση αυτού του κεφαλαίου, θα γνωρίζετε αρκετά ώστε να ξεκινήσετε να γράφετε παιχνίδι με πίνακα βαθμολογίας και αυξανόμενο βαθμό δυσκολίας όσο ο παίκτης ανεβαίνει επίπεδα.

Προσθήκη του κουμπιού Play

Σε αυτή την ενότητα, θα προσθέσουμε το κουμπί Play το οποίο θα εμφανίζεται πριν από την έναρξη ενός παιχνιδιού και θα ξαναεμφανίζεται όταν το παιχνίδι τελειώσει έτσι ώστε να δίνει τη δυνατότητα στον παίκτη να παίξει ξανά. Αυτή τη στιγμή το παιχνίδι ξεκινά μόλις τρέξουμε το alien_invasion.py. Ας ξεκινήσουμε το παιχνίδι σε ανενεργή κατάσταση και έπειτα να δώσουμε την προτροπή στον παίκτη να πατήσει το κουμπί Play για να ξεκινήσει να παίζει. Για το κάνουμε αυτό, θα πρέπει να τροποποιήσουμε τη μέθοδο __init__() της GameStats:

game_stats.py

```
def __init__(self, ai_game):
    """Initialize statistics."""
    self.settings = ai_game.settings
```

```
        self.reset_stats()

        # Start game in an inactive state.
        self.game_active = False
```

Τώρα το παιχνίδι θα ξεκινά σε ανενεργή κατάσταση και ο παίκτης θα μπορεί να ξεκινήσει να παίζει μόνο όταν πατήσει το κουμπί Play.

Δημιουργία κλάσης Button

Επειδή η Pygame δεν έχει ενσωματωμένη μέθοδο για τη δημιουργία κουμπιών, θα γράψουμε μια κλάση Button για να δημιουργήσουμε ένα γεμάτο ορθογώνιο με μια ετικέτα. Μπορούμε να χρησιμοποιήσουμε αυτόν τον κώδικα για να φτιάξουμε οποιοδήποτε κουμπί στο παιχνίδι. Εδώ έχουμε το πρώτο μέρος της κλάσης Button το οποίο αποθηκεύουμε ως button.py:

button.py

```
import pygame.font

class Button:

❶     def __init__(self, ai_game, msg):
          """Initialize button attributes."""
          self.screen = ai_game.screen
          self.screen_rect = self.screen.get_rect()

          # Set the dimensions and properties of the button.
❷         self.width, self.height = 200, 50
          self.button_color = (0, 255, 0)
          self.text_color = (255, 255, 255)
❸         self.font = pygame.font.SysFont(None, 48)

          # Build the button's rect object and center it.
❹         self.rect = pygame.Rect(0, 0, self.width, self.height)
          self.rect.center = self.screen_rect.center

          # The button message needs to be prepped only once.
❺         self._prep_msg(msg)
```

Πρώτα, εισάγουμε το δομοστοιχείο pygame.font, το οποίο επιτρέπει στην Pygame να παράγει κείμενο στην οθόνη. Η μέθοδος __init__() λαμβάνει τις παραμέτρους self, το αντικείμενο ai_game, και το msg, που περιέχει το κείμενο του κουμπιού ❶. Ορίζουμε τις διαστάσεις του κουμπιού ❷, κι έπειτα ορίζουμε την button_color να χρωματίσει το αντικείμενο rect του κουμπιού ένα ανοιχτό πράσινο, και τέλος, ορίζουμε την text_color να παράγει κείμενο σε λευκό χρώμα.

Στο ❸, ετοιμάζουμε μια ιδιότητα font για να παράγουμε κείμενο. Το όρισμα None λέει στην Pygame να δημιουργήσει μια προεπιλεγμένη γραμματοσειρά και το 48 προσδιορίζει το μέγεθος του κειμένου. Για να τοποθετήσουμε το κουμπί στο κέντρο της οθόνης, δημιουργούμε ένα rect για το κουμπί ❹ και ορίζουμε την ιδιότητα center του

ώστε να ταιριάζει με αυτή της οθόνης.

Η Pygame διαχειρίζεται κείμενο παράγοντας τη συμβολοσειρά που θέλουμε να εμφανίσουμε ως εικόνα. Στο ❺, καλούμε την _prep_msg() για να διαχειριστεί αυτή την παραγωγή.

Ο κώδικας για την _prep_msg() έχει ως εξής:

button.py

```
    def _prep_msg(self, msg):
        """Turn msg into a rendered image and center text on the
button."""
❶       self.msg_image = self.font.render(msg, True, self.text_color,
                self.button_color)
❷       self.msg_image_rect = self.msg_image.get_rect()
        self.msg_image_rect.center = self.rect.center
```

Η μέθοδος _prep_msg() χρειάζεται μια παράμετρο self και το κείμενο που θα δοθεί ως εικόνα (msg). Η κλήση στην font.render() μετατρέπει το κείμενο που είναι αποθηκευμένο σε msg σε εικόνα, την οποία έπειτα αποθηκεύουμε στο self.msg_image ❶. Η μέθοδος font.render() λαμβάνει επίσης μια τιμή Boolean για την αύξηση ή μείωση της εξομάλυνσης (η εξομάλυνση κάνει τις γωνίες του κειμένου πιο ομαλές). Τα υπόλοιπα ορίσματα είναι το χρώμα της γραμματοσειράς και το χρώμα του φόντου. Ορίζουμε την εξομάλυνση σε True και το φόντο του κειμένου στο ίδιο χρώμα με το κουμπί. (Αν δεν συμπεριλάβετε χρώμα φόντου, η Pygame θα προσπαθήσει να παράγει τη γραμματοσειρά με ένα διαφανές φόντο).

Στο ❷, τοποθετούμε την εικόνα κειμένου στο κέντρο του κουμπιού δημιουργώντας ένα rect από την εικόνα και ορίζοντας την ιδιότητα center του να ταιριάζει με αυτή του κουμπιού.

Τέλος, δημιουργούμε μια μέθοδο draw_button() την οποία μπορούμε να καλέσουμε για να εμφανίσουμε το κουμπί στην οθόνη:

button.py

```
    def draw_button(self):
        # Draw blank button and then draw message.
        self.screen.fill(self.button_color, self.rect)
        self.screen.blit(self.msg_image, self.msg_image_rect)
```

Καλούμε την screen.fill() να σχεδιάσει το ορθογώνιο μέρος του κουμπιού. Κι έπειτα καλούμε την screen.blit() να σχεδιάσει την εικόνα κειμένου στην οθόνη, περνώντας σε αυτή μια εικόνα και το αντικείμενο rect που σχετίζεται με την εικόνα. Έτσι ολοκληρώνεται η κλάση Button.

Σχεδιασμός του κουμπιού στην οθόνη

Θα χρησιμοποιήσουμε την κλάση Button για να δημιουργήσουμε ένα κουμπί Play στο AlienInvasion. Πρώτα, ενημερώνουμε τις εντολές import: