Η Μέρα της Εβδομάδας

4

29 Οκτωβρίου 2015
22:07

Στο κεφάλαιο αυτό θ' αναπτύξουμε ένα πρόγραμμα για να υπολογίζουμε σε ποια ημέρα της εβδομάδας αντιστοιχεί μια συγκεκριμένη ημερομηνία. Θα δανειστούμε τη μέθοδο υπολογισμού από τον Gauss. Στην πορεία θα έχουμε την ευκαιρία να εξασκηθούμε σε όλα τα προγραμματιστικά εργαλεία που έχουμε συναντήσει μέχρι τώρα.

Έννοιες: δομή επιλογής, δομή επανάληψης, υποπρογράμματα, πλειάδες

Η επίλυση ημερολογιακών και αστρονομικών προβλημάτων ήταν ένα από τα πρώτα πεδία εφαρμογής υπολογιστικών μεθόδων. Η μέτρηση του χρόνου και η πρόβλεψη των κινήσεων των ουράνιων σωμάτων ήταν προβλήματα εξαιρετικής σημασίας καθ' όλη την ιστορία της ανθρωπότητας, με πρακτικές εφαρμογές αλλά κι επιπτώσεις πολιτικές, επιστημονικές και φιλοσοφικές. Η ύπαρξη του μηχανισμού των Αντικυθήρων δείχνει από πόσο νωρίς επιδιώχθηκε η αυτοματοποίηση τέτοιου είδους υπολογισμών.

Carl Friedrich Gauss (1777-1855)

Ο ιδιοφυής μαθηματικός που ασχολήθηκε με αναρίθμητα προβλήματα σε πολλά διαφορετικά πεδία.

Το να υπολογίσει κανείς την ημέρα της εβδομάδας για μια συγκεκριμένη ημερομηνία είναι ένα καλά μελετημένο πρόβλημα. Για την επίλυση του προβλήματος υπάρχουν πολλές προσεγγίσεις. Μερικές βασίζονται σε μνημονικούς κανόνες, με αποτέλεσμα ορισμένοι άνθρωποι να είναι σε θέση να λύσουν το πρόβλημα μέσα σε μερικά δευτερόλεπτα, κάνοντας υπολογισμούς μόνο με το μυαλό τους. Εμείς θα χρησιμοποιήσουμε μια μέθοδο του Gauss, η οποία είναι απλούστατη στην υλοποίησή της, αν και ο τρόπος με τον οποίο λειτουργεί δεν είναι προφανής.

Πες Μου Πότε

Ας ξεκινήσουμε ρωτώντας το χρήστη ποια είναι η ημερομηνία για την οποία θα ήθελε να μάθει την αντίστοιχη ημέρα της εβδομάδας.

```
# είσοδος ημερομηνίας από το χρήστη
print("Έτος: ", end="")
year = int(input())
print("Μήνας: ", end="")
month = int(input())
print("Ημέρα: ", end="")
day = int(input())
```

Εναλλακτικά:

```
# είσοδος ημερομηνίας από το χρήστη
year = int(input("Έτος: "))
month = int(input("Μήνας: "))
day = int(input("Ημέρα: "))
```

Τί Μέρα Έχουμε Σήμερα;

Και τώρα που έχουμε την ημερομηνία πώς θα υπολογίσουμε το ζητούμενο; Ποιά είναι η μέθοδος του Gauss;

Οι γραμμές που ακολουθούν είναι μια υλοποίηση της μεθόδου του Gauss. Πρόκειται για εντολές που υπολογίζουν την τιμή ορισμένων αριθμητικών παραστάσεων, για να μας δώσουν τελικά το αποτέλεσμα που χρειαζόμαστε: έναν ακέραιο από το 0 μέχρι και το 6, που αντιστοιχεί στις ημέρες από την Κυριακή μέχρι και το Σάββατο. Ο τρόπος που λειτουργεί η μέθοδος δεν είναι προφανής, αλλά εδώ μας ενδιαφέρει να τη χρησιμοποιήσουμε, όχι να την αναλύσουμε.

```
8  # υπολογισμός ημέρας με την μέθοδο του Gauss
9  # προσαρμογή μήνα και έτους 2 μήνες προς τα πίσω
10 if month > 2:
11     month = month - 2
12 else:
13     month = month + 10
14     year = year - 1
15 # όρος για δίσεκτα έτη
16 leap = 5*(year%4) + 4*(year%100) + 6*(year%400)
17 # υπολογισμός ημέρας της εβδομάδας
18 weekday = (day + leap + int(2.6 * month - 0.2)) % 7
```

Μπορείτε να ελέγξετε αν οι εντολές λειτουργούν σωστά κάνοντας μια δοκιμή με την σημερινή ημερομηνία. Δεν θα πρέπει βέβαια να παραλείψουμε την εμφάνιση του αποτελέσματος στην οθόνη.

```
19 # εμφάνιση ημέρας της εβδομάδας (0:Κυριακή - 6:Σάββατο)
20 print("Ημέρα της εβδομάδας:", weekday)
```

src/weekday.1.py

Κλεισμένη στο Κουτάκι

Αυτή η μέθοδος είναι εντυπωσιακή: υπολογίζει το αποτέλεσμα με λίγες πράξεις, αν και το πρόβλημα είναι περίπλοκο. Ωστόσο με ενοχλεί να βλέπω μέσα στο πρόγραμμα αυτόν τον ακατάληπτο κώδικα.

Μπορούμε να απομονώσουμε τον ομολογουμένως δυσνόητο υπολογισμό της ημέρας της εβδομάδας μέσα σ' ένα υποπρόγραμμα. Αυτό θα δέχεται μια τριάδα παραμέτρων (ημέρα, μήνας, έτος) που αντιστοιχούν σε μια ημερομηνία και θα επιστρέφει έναν ακέραιο από το 0 μέχρι και το 6, που αντιστοιχεί σε μια ημέρα της εβδομάδας.

```
def computeWeekday(day, month, year):
    """ Υπολογίζει με την μέθοδο του Gauss
    την ημέρα της εβδομάδας στην οποία αντιστοιχεί
    μια συγκεκριμένη ημερομηνία.
    day, month, year: ορίζει την ημερομηνία
    """
    # προσαρμογή μήνα και έτους 2 μήνες προς τα πίσω
    if month > 2:
        month = month - 2
    else:
        month = month + 10
        year = year - 1
    # όρος για δίσεκτα έτη
    leap = 5*(year%4) + 4*(year%100) + 6*(year%400)
    # υπολογισμός κι επιστροφή ημέρας της εβδομάδας
    return (day + leap + int(2.6 * month - 0.2)) % 7
```

Ο κώδικας της μεθόδου δεν έχει αλλάξει, αλλά πλέον είναι «κλεισμένος» μέσα στο υποπρόγραμμα. Όποιος θέλει να το χρησιμοποιήσει δεν χρειάζεται να καταλαβαίνει πως λειτουργεί. Δεν χρειάζεται καν να γνωρίζει με ποια μέθοδο υπολογίζεται το αποτέλεσμα.

Στο κύριο πρόγραμμα, απλά καλούμε την συνάρτηση, δίνοντας ως παραμέτρους τις τιμές που διαβάσαμε από το χρήστη.

```
# είσοδος ημερομηνίας από το χρήστη
print("Έτος: ", end="")
year = int(input())
print("Μήνας: ", end="")
month = int(input())
print("Ημέρα: ", end="")
day = int(input())
# κλήση υποπρογράμματος για υπολογισμό ημέρας
weekday = computeWeekday(day, month, year)
```

src/weekday.2.py

Στις 32 Του Μηνός

Ο χρήστης μπορεί να δώσει αυθαίρετους αριθμούς για το έτος, τον μήνα και την ημέρα. Δεν χρειάζεται κάποιος έλεγχος;

Είναι απαραίτητο η τιμή που θα δοθεί για το μήνα να είναι ανάμεσα στο 1 και το 12. Επίσης, είναι απαραίτητο η τιμή που θα δοθεί για την ημέρα να είναι ανάμεσα στο 1 και το πλήθος των ημερών του μήνα. Για το έτος υπάρχει περιθώριο ελαστικότητας, αλλά και πάλι θα ήταν καλό να περιορίσουμε την τιμή σε λογικά όρια.

Ας εξετάσουμε αρχικά πως θα μπορούσαμε να ελέγξουμε ότι η τιμή που δίνεται για τον μήνα είναι έγκυρη. Πιθανώς το πρώτο πράγμα που θα έγραφε κανείς να ήταν το εξής:

```
print("Μήνας: ", end="")
month = int(input())
if month < 1 or month > 12:
    # μήνυμα λάθους
    print("Δώστε τιμή από 1 μέχρι 12")
```

Όμως αυτό δεν αρκεί γιατί, σε περίπτωση μη-έγκυρης τιμής, ο χρήστης θα πρέπει και πάλι να πληκτρολογήσει μια τιμή για τον μήνα, η οποία θα πρέπει και πάλι να ελεγχθεί και, πιθανώς, να μην είναι ούτε αυτή την φορά έγκυρη, κ.ο.κ. Με άλλα λόγια, η ανάγνωση τιμής και ο έλεγχός της θα πρέπει να *επαναλαμβάνονται*, μέχρι να διαβαστεί έγκυρη τιμή. Θα χρειαστούμε λοιπόν μια *δομή επανάληψης*.

```
# επανάληψη, όσο η τιμή είναι εκτός ορίων
while True:
    print("Μήνας: ", end="")
    month = int(input())
    if month < 1 or month > 12:
        # μήνυμα λάθους
        print("Δώστε τιμή από 1 μέχρι 12")
    else:
        break
```

Εδώ ο χρήστης πρακτικά εγκλωβίζεται σ' έναν κύκλο. Η ανάγνωση τιμής (μαζί με το σχετικό μήνυμα λάθους) επαναλαμβάνεται μέχρι ο χρήστης να παρέχει μια αποδεκτή τιμή. Μια εναλλακτική προσέγγιση της ίδιας ιδέας είναι η εξής:

```
20 print("Μήνας: ", end="")
21 month = int(input())
22 # επανάληψη, όσο η τιμή είναι εκτός ορίων
23 while month < 1 or month > 12:
24     # μήνυμα λάθους
25     print("Δώστε τιμή από 1 μέχρι 12")
26     print("Μήνας: ", end="")
27     month = int(input())
```

src/weekday.3.py

Και σε πολλά άλλα προγράμματα θα χρειαστεί να διαβάσουμε από το χρήστη τιμές, εξασφαλίζοντας ότι οι τιμές αυτές είναι *έγκυρες*, δηλαδή ικανοποιούν κάποιους περιορισμούς.

Μια Απ' Τα Ίδια;

Μου φαίνεται πως αυτό ακριβώς που κάναμε για το μήνα θα πρέπει να το κάνουμε για το έτος και για την ημέρα (αλλά με διαφορετικά όρια). Πάλι τα ίδια θα γράφουμε;

Και για τις τρεις τιμές (έτος, μήνας, ημέρα) έχουμε το ίδιο ακριβώς πρόβλημα: θέλουμε να εμφανίζεται στο χρήστη μια *προτροπή* και στη συνέχεια να διαβάζεται από το χρήστη μια ακέραια τιμή που θα πρέπει να βρίσκεται εντός συγκεκριμένων ορίων.

Θα λύσουμε αυτό το πρόβλημα με γενικό τρόπο, κατασκευάζοντας ένα *υποπρόγραμμα* που δέχεται σαν παραμέτρους την προτροπή και τα όρια κι επιστρέφει την τιμή που διαβάστηκε από το χρήστη. Ουσιαστικά θα γενικεύσουμε τον κώδικα που αναπτύξαμε λίγο πριν για τον έλεγχο της τιμής του μήνα.

```
17 def readInt(prompt,lower,upper):
18     """ Εμφανίζει μια προτροπή και μετά διαβάζει
19     από το χρήστη κι επιστρέφει έναν ακέραιο αριθμό,
20     εξασφαλίζοντας ότι βρίσκεται εντός ορίων.
21     prompt: η προτροπή που εμφανίζεται στο χρήστη
22     lower, upper: τα όρια
23     """
24     # εμφάνιση προτροπής και ανάγνωση τιμής
25     print(prompt, end="")
26     num = int(input())
27     # επανάληψη, όσο η τιμή είναι εκτός ορίων
28     while num < lower or num > upper:
29         # μήνυμα λάθους
30         print("Δώστε τιμή από",lower,"μέχρι",upper)
31         # εμφάνιση προτροπής και ανάγνωση τιμής
32         print(prompt, end="")
33         num = int(input())
34     # η έγκυρη τιμή επιστρέφεται
35     return num
```

Θα χρησιμοποιήσουμε αυτό το υποπρόγραμμα τρεις φορές, καλώντας το κάθε φορά με διαφορετικές παραμέτρους, για να εισάγουμε τιμές για το έτος, το μήνα και την ημέρα.

```
36 # είσοδος ημερομηνίας από το χρήστη
37 year = readInt("Έτος: ", 1923, 2999)
38 month = readInt("Μήνας: ", 1, 12)
39 day = readInt("Ημέρα: ", 1, 31)
40 # κλήση υποπρογράμματος για υπολογισμό ημέρας
41 weekday = computeWeekday(day, month, year)
```

src/weekday.4.py

Το 1923 είναι η χρονιά που υιοθετήθηκε η χρήση του Γρηγοριανού ημερολογίου στην Ελλάδα. Το άνω όριο είναι αυθαίρετο.

Αναπτύξαμε αυτό το υποπρόγραμμα γιατί ήταν απαραίτητο σε τρία διαφορετικά σημεία του προγράμματός μας να διαβάζουμε έναν ακέραιο αριθμό εντός συγκεκριμένων ορίων. Αντί να αντιγράψουμε τις ίδιες εντολές σε τρία σημεία, τις τοποθετήσαμε μέσα στο υποπρόγραμμα για να τις καλούμε όποτε τις χρειαζόμαστε.

Όμως το υποπρόγραμμα που αναπτύξαμε δεν λειτουργεί μόνο για έτη, μήνες και ημέρες, αλλά μπορεί να χρησιμοποιηθεί και σε οποιαδήποτε άλλη περίπτωση θέλουμε να διαβάσουμε από το χρήστη έναν ακέραιο αριθμό εντός ορίων. Με άλλα λόγια, το υποπρόγραμμα λύνει το πρόβλημα με γενικό τρόπο και είναι επαναχρησιμοποιήσιμο.

Πόσες Έχει Ο Μήνας;

Δεν έχουν όμως όλοι οι μήνες 31 ημέρες. Αφού τον κάνουμε τον έλεγχο, ας τον κάνουμε σωστά.

Στο σημείο αυτό χρειάζεται να υπολογίσουμε το πλήθος των ημερών ενός μήνα. Όπως φαντάζεστε, θα λύσουμε αυτό το υποπρόβλημα κατασκευάζοντας το αντίστοιχο υποπρόγραμμα, ένα ακόμα "εξάρτημα" που κάνει μια συγκεκριμένη δουλειά: δέχεται σαν παράμετρο το μήνα και το έτος κι επιστρέφει τον αριθμό των ημερών του μήνα.

Οι κανόνες που καθορίζουν πόσες ημέρες έχει ο κάθε μήνας είναι οι εξής: πριν τον Αύγουστο, οι άρτιοι μήνες έχουν 30 ημέρες και οι περιττοί έχουν 31, ενώ από τον Αύγουστο και μετά ισχύει το αντίστροφο. Εξαίρεση αποτελεί ο Φλεβάρης που έχει 28 ημέρες, εκτός κι αν το έτος είναι δίσεκτο, οπότε έχει 29.

Από τα παραπάνω φαίνεται πως τίθεται ένα ακόμα μικρότερο πρόβλημα: χρειάζεται να γνωρίζουμε αν ένα έτος είναι δίσεκτο ή όχι. Ας ξεκινήσουμε λοιπόν με την επίλυση αυτού του υποπροβλήματος, κατασκευάζοντας το αντίστοιχο υποπρόγραμμα, το οποίο θα δέχεται σαν παράμετρο ένα έτος και θα αποφαίνεται αν είναι δίσεκτο ή όχι. Εξ' ορισμού, δίσεκτα είναι τα έτη που διαιρούνται με το 4, αλλά όχι με το 100, καθώς και τα έτη που διαιρούνται με το 400.

```
def isLeap(year):
    """ Επιστρέφει την τιμή True αν το έτος year
    είναι δίσεκτο, αλλιώς επιστρέφει False.
    """
    if (year % 4 == 0 and year % 100 > 0) or
      year % 400 == 0:
        return True
    else:
        return False
```

Στην πραγματικότητα, η δομή επιλογής δεν είναι απαραίτητη. Ορθότερη προσέγγιση είναι η εξής:

```
def isLeap(year):
    """ Επιστρέφει την τιμή True αν το έτος year
    είναι δίσεκτο, αλλιώς επιστρέφει False.
    """
    return ((year % 4 == 0 and year % 100 > 0) or
            year % 400 == 0)
```

Η τιμή μιας συνθήκης είναι είτε αληθής (`True`), είτε ψευδής (`False`). Εδώ λοιπόν επιστρέφεται απευθείας η τιμή της συνθήκης που ελέγχει αν το έτος `year` είναι δίσεκτο.

Τώρα μπορούμε να υλοποιήσουμε το υποπρόγραμμα που υπολογίζει τον αριθμό των ημερών ενός μήνα. Το έτος είναι απαραίτητο σαν παράμετρος στο υποπρόγραμμα γιατί ο μήνας μπορεί να είναι ο Φλεβάρης, οπότε το έτος επηρρεάζει τον αριθμό των ημερών.

```
42 def daysOfMonth(m,y):
43     """ Επιστρέφει το πλήθος των ημερών ενός μήνα.
44     m: ο αριθμός του μήνα (1-12)
45     y: ο αριθμός του έτους (απαραίτητο όταν m=2)
46     """
47     if m == 2:
48         # Φεβρουάριος
49         if isLeap(y):
50             return 29
51         else:
52             return 28
53     elif m <= 7:
54         # Ιανουάριος - Ιούλιος (πλην Φεβρουαρίου)
55         if m % 2 == 1:
56             return 31
57         else:
58             return 30
59     else:
60         # Αύγουστος - Δεκέμβριος
61         if m % 2 == 1:
62             return 30
63         else:
64             return 31
```

Η ανάγνωση τιμής για την ημέρα μπορεί πλέον να γίνει καθορίζοντας σωστά το πλήθος των ημερών του μήνα.

```
65 # είσοδος ημερομηνίας από το χρήστη
66 year = readInt("Έτος: ", 1923, 2999)
67 month = readInt("Μήνας: ", 1, 12)
68 day = readInt("Ημέρα: ", 1, daysOfMonth(month,year))
```

src/weekday.5.py

Από τα παραπάνω βγαίνει στην επιφάνεια το μεγαλύτερο πλεονέκτημα που προκύπτει από τη χρήση υποπρογραμμάτων. Είμαστε "αναγκασμένοι" να αναλύουμε τα προβλήματα και να τ' αντιμετωπίζουμε *τμηματικά*. Κάθε φορά εστιάζουμε την προσοχή μας σε μικρότερα κι απλούστερα κομμάτια του γενικού προβλήματος. Στη συνέχεια, συνθέτουμε τη λύση, συναρμολογώντας τα κομμάτια αυτά.

Και Κάτι Ακόμα

Δεν μου αρέσει που το πρόγραμμα εμφανίζει έναν αριθμό και πρέπει ο χρήστης να σκέφτεται την αντιστοιχία με τις ημέρες.

Υπάρχουν αρκετοί τρόποι ν' αντιστοιχίσουμε τους ακέραιους από το 0 μέχρι και το 6 με τις ημέρες από την Κυριακή μέχρι και το Σάββατο. Ένας από αυτούς είναι να τοποθετήσουμε τα ονόματα των ημερών της εβδομάδας μέσα σε μια *πλειάδα*. Η πλειάδα είναι μια λίστα τιμών *που δεν μπορεί να τροποποιηθεί*.

```
# πλειάδα με τα ονόματα των ημερών
dayNames = ('Κυρ','Δευ','Τρι','Τετ','Πεμ','Παρ','Σαβ')
```

Είναι η πρώτη φορά που ένα όνομα, το `dayNames`, δεν αντιστοιχεί σε μια μεμονωμένη τιμή αλλά σε μια *συλλογή τιμών*, όπως είναι εδώ η πλειάδα με τα ονόματα των ημερών.

Η σειρά των ημερών μέσα στην πλειάδα έχει σημασία γιατί κάθε θέση είναι αριθμημένη, με την αρίθμηση να ξεκινάει από το μηδέν. Μπορούμε λοιπόν να χρησιμοποιήσουμε την (ακέραια) ημέρα της εβδομάδας που έχουμε υπολογίσει για να έχουμε πρόσβαση στην αντίστοιχη θέση της πλειάδας που περιέχει το όνομα της ημέρας.

Το πρώτο στοιχείο της πλειάδας είναι το `dayNames[0]`, με τιμή `'Κυρ'`, το δεύτερο είναι το `dayNames[1]`, με τιμή `'Δευ'`, ενώ το τελευταίο είναι το `dayNames[6]`, με τιμή `'Σαβ'`. Η αρίθμηση μπορεί να είναι κι αντίστροφη, με το πρώτο στοιχείο από το τέλος να είναι το `dayNames[-1]`, με τιμή `'Σαβ'`, κ.ο.κ.

```
69 # πλειάδα με τα ονόματα των ημερών
70 dayNames = ('Κυρ','Δευ','Τρι','Τετ','Πεμ','Παρ','Σαβ')
71 # κλήση υποπρογράμματος για υπολογισμό ημέρας
72 weekday = computeWeekday(day, month, year)
73 # εμφάνιση ημέρας της εβδομάδας (0:Κυριακή - 6:Σάββατο)
74 print("Ημέρα της εβδομάδας:", dayNames[weekday])
```

src/weekday.6.py

Πλήρες Τελικό Πρόγραμμα

```
1  def computeWeekday(day, month, year):
2      """ Υπολογίζει με την μέθοδο του Gauss
3      την ημέρα της εβδομάδας στην οποία αντιστοιχεί
4      μια συγκεκριμένη ημερομηνία.
5      day, month, year: ορίζει την ημερομηνία
6      """
7      # προσαρμογή μήνα και έτους 2 μήνες προς τα πίσω
8      if month > 2:
9          month = month - 2
10     else:
11         month = month + 10
12         year = year - 1
13     # όρος για δίσεκτα έτη
14     leap = 5*(year%4) + 4*(year%100) + 6*(year%400)
15     # υπολογισμός κι επιστροφή ημέρας της εβδομάδας
16     return (day + leap + int(2.6 * month - 0.2)) % 7
```

...συνεχίζεται στην επόμενη σελίδα.

```
def readInt(prompt,lower,upper):
    """ Εμφανίζει μια προτροπή και μετά διαβάζει
    από το χρήστη κι επιστρέφει έναν ακέραιο αριθμό,
    εξασφαλίζοντας ότι βρίσκεται εντός ορίων.
    prompt: η προτροπή που εμφανίζεται στο χρήστη
    lower, upper: τα όρια
    """
    # εμφάνιση προτροπής και ανάγνωση τιμής
    print(prompt, end="")
    num = int(input())
    # επανάληψη, όσο η τιμή είναι εκτός ορίων
    while num < lower or num > upper:
        # μήνυμα λάθους
        print("Δώστε τιμή από",lower,"μέχρι",upper)
        # εμφάνιση προτροπής και ανάγνωση τιμής
        print(prompt, end="")
        num = int(input())
    # η έγκυρη τιμή επιστρέφεται
    return num

def isLeap(year):
    """ Επιστρέφει την τιμή True αν το έτος year
    είναι δίσεκτο, αλλιώς επιστρέφει False.
    """
    return ((year % 4 == 0 and year % 100 > 0) or
            year % 400 == 0)

def daysOfMonth(m,y):
    """ Επιστρέφει το πλήθος των ημερών ενός μήνα.
    m: ο αριθμός του μήνα (1-12)
    y: ο αριθμός του έτους (απαραίτητο όταν m=2)
    """
    if m == 2:
        # Φεβρουάριος
        if isLeap(y):
            return 29
        else:
            return 28
    elif m <= 7:
        # Ιανουάριος - Ιούλιος (πλην Φεβρουαρίου)
        if m % 2 == 1:
            return 31
        else:
            return 30
    else:
        # Αύγουστος - Δεκέμβριος
        if m % 2 == 1:
            return 30
        else:
            return 31
```

```
65 # είσοδος ημερομηνίας από το χρήστη
66 year = readInt("Έτος: ", 1923, 2999)
67 month = readInt("Μήνας: ", 1, 12)
68 day = readInt("Ημέρα: ", 1, daysOfMonth(month,year))
69 # κλήση υποπρογράμματος για υπολογισμό ημέρας
70 weekday = computeWeekday(day, month, year)
71 # εμφάνιση ημέρας της εβδομάδας (0:Κυριακή - 6:Σάββατο)
72 print("Ημέρα της εβδομάδας:", weekday)
```

src/weekday.final.py

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ *Οργανωμένες συλλογές τιμών όπως οι πλειάδες ονομάζονται* δομές δεδομένων. *Μια δομή δεδομένων δεν είναι ένα απλό σύνολο από τιμές, δεν αποτελεί μια απλή γενίκευση των μεταβλητών, αλλά είναι δομημένη με συγκεκριμένο τρόπο. Τα διαφορετικά είδη δομών δεδομένων διαφοροποιούνται κυρίως από τον τρόπο με τον οποίο τα δεδομένα αποθηκεύονται, ανακτώνται και συσχετίζονται μεταξύ τους.*
