

Συμπληρωματικές Ασκήσεις 4

Επιμέλεια : Αλέξανδρος Αργυρόπουλος

Επικοινωνία: alex.argiropoulos2004@gmail.com

1. Να υλοποιηθεί σε κώδικα η εξής μετατροπή :

$$l1 = [3, [5, 6, 7], [1, 1]] \mapsto l2 = [[], 15, [5, 6, 7, 8], [1, 1], [1, 1], True]$$

Διατηρώντας την αρχική λίστα $l1$ και δημιουργώντας μια νέα $l2$. Τι επιστρέφουν οι κλήσεις της len στις εξής περιπτώσεις:

α) $\text{len}(l1)$, β) $\text{len}(l2)$, γ) $\text{len}(l1[2])$ δ) $\text{len}(l1[0])$, ε) $\text{len}(l2[-4])$, στ) $\text{len}(l1[1][1])$.

Τι θα εμφανίσει ο εξής κώδικας :

```
l1 = l2[-1: :-6]
```

```
if l1[ len(l2) - 15 // 7 - 9 % 5 ] :
```

```
    print( l1 + [is_prime(29) and is_perfect(28)] ) # Συναρτήσεις ελέγχου πρώτων και τέλειων αντίστοιχα
```

```
else:
```

```
    print(l1.append(2**5))
```

```
print(l1, l2)
```

2. **Το τρίγωνο του Pascal** : Να φτιαχτεί πρόγραμμα που θα αποτελείται από μία συνάρτηση $\text{main}()$: η οποία θα δέχεται ως είσοδο έναν αριθμό $n \in \mathbb{N}^+$, που θα δείχνει έως ποια γραμμή του τριγώνου θα φτάσει ο υπολογισμός. Επιπλέον θα ζητάει μία τιμή k στο $[n] = \{1, \dots, n\}$, μέχρι αυτό να γίνει σωστά. Η έξοδος της συνάρτησης θα είναι το τρίγωνο του Pascal έως τη γραμμή που ζητήθηκε και επίσης το άθροισμα που προκύπτει για τον υπολογισμό του $(x + y)^k$. Φυσικά θα πρέπει να υπάρχει και μία συνάρτηση $\text{Pascal}(n)$: η οποία θα επιστρέφει το τρίγωνο έως την n -οστή γραμμή.
- Επέκταση: Η συνάρτηση Pascal να φτιαχτεί σε δύο εκδοχές μία που θα χρησιμοποιεί υπολογισμό διωνυμικού για την εύρεση κάθε στοιχείου και μία που θα χρησιμοποιεί ότι κάθε μη μοναδιαίο στοιχείο προκύπτει ως άθροισμα των δύο "πάνω".

					1					
				1		1				
			1		2		1			
		1		3		3		1		
	1		4		6		4		1	
1		5		10		10		5		1

Είσοδος:

Γραμμές : $n = 6$ και $k = 4$

Έξοδος:

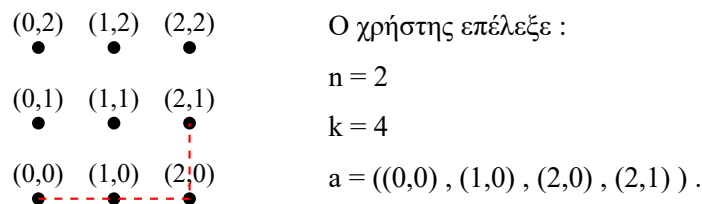
1) $x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^4$

Σχόλιο: Αφού δεν αναφέρεται συγκεκριμένος τρόπος εκτύπωσης του τριγώνου, μπορεί να γίνει και αλλιώς π.χ. μετατοπίζοντας τις $n-1$ γραμμές, του δοθέντος σχήματος, προς τα αριστερά δημιουργώντας ένα "ορθογώνιο".

Να γραφεί σε κώδικα Latex (έχοντας δεδομένα τα απαραίτητα πακέτα) η δεύτερη έκδοση του προγράμματος για :

$$n \geq k = 2, \quad x = \lim_{M \rightarrow \infty} \int_{\pi}^M \frac{(-1)^{\nu}}{x^2} dx, \quad y = \sum_{\gamma=0}^{\lambda} \begin{pmatrix} \cos \alpha \\ \gamma \end{pmatrix}$$

3. Να φτιαχτεί ένα πρόγραμμα που θα ζητάει από τον χρήστη δύο θετικούς ακέραιους αριθμούς n, k . Με τον n θα περιοριζόμαστε στο ακέραιο επίπεδο από $(0,0)$ έως και (n,n) , δηλαδή όλα τα (i,j) , $0 \leq i, j \leq n$. Έπειτα θα ζητάει από τον χρήστη μία ακολουθία a_{ν} k σημείων, η οποία θα αποτελεί ένα μονοπάτι (Υπόδειξη: αν $a_{\nu} = (b, c)$, $a_{\nu+1} = (d, e)$ τότε $|d - b|, |e - c| \leq 1$). Το πρόγραμμα θα εκτυπώνει τα σημεία του επιπέδου που δημιουργήθηκε ως τελείες και το μονοπάτι, ενώνοντας τα σημεία με $—$ (3^+ παύλες) ή με $|$.



4. Χρησιμοποιώντας το module random να φτιάξετε ένα πρόγραμμα το οποίο θα δημιουργεί μία δισδιάστατη λίστα, τα στοιχεία της οποίας θα είναι n (δίνεται από τον χρήστη) λίστες τυχαίου μήκους και τυχαίων τιμών. Το μέγιστο τυχαίο μήκος len_max των λιστών θα πρέπει, επίσης, να δίνεται από τον χρήστη και οι τιμές να κυμαίνονται μεταξύ του 0 και του len_max^2 . Το πρόγραμμα, ακόμη, θα επεξεργάζεται τη δισδιάστατη λίστα και θα τη διατάσσει κατά αύξουσα σειρά (όχι αναγκαστικά γνησίως) ως προς το μήκος των λιστών της. Τέλος θα διατάσσει, χωρίς χρήση κάποιας έτοιμης συνάρτησης ή μεθόδου, κατά φθίνουσα σειρά κάθε λίστα και θα εκτυπώνει την τελική έκδοση της δισδιάστατης λίστας.
5. **Το πρόβλημα των ταιριασμάτων** : Έστω μία δισδιάστατη λίστα l μήκους n , με $l[i] = [i+1, \text{"symbol"}]$, $\text{symbol} \in \{!, \&, \$, ^\}$, $\forall i = 0, \dots, n-1$ και η επιλογή του συμβόλου γίνεται τυχαία. Να φτιαχτεί μία συνάρτηση `shuffle_inf` που θα επιστρέφει την κατάλληλη "πληροφορία" για μία τυχαία εναλλαγή των θέσεων τόσο των τιμών όσο και των συμβόλων (οι αλλαγές γίνονται στα πλαίσια των στηλών : παραμένει δηλαδή η δομή [τιμή, σύμβολο]). (Μπορεί να φτιαχτεί και μία συνάρτηση `shuffle` που χρησιμοποιεί αυτή την πληροφορία και επιστρέφει την ανακατεμένη λίστα, χωρίς να επηρεαστεί η αρχική). Τώρα σε κάθε γραμμή έχουμε τις εξής περιπτώσεις : α) **ταίριασμα** (τ) : Η γραμμή της αρχικής ταυτίζεται με την αντίστοιχή της στην ανακατεμένη. β) **μερικό ταίριασμα** ($\mu\tau$) : Ένα από τα δύο στοιχεία ταυτίζεται με το αντίστοιχό του στην ανακατεμένη και τέλος γ) **κανένα ταίριασμα** ($\kappa\tau$). Να φτιαχτεί συνάρτηση `matches` που θα επιστρέφει σε λίστα το πλήθος των τ , $\mu\tau$, $\kappa\tau$. Τέλος να αξιοποιήσετε κατάλληλα το πρόγραμμα ώστε να προσομοιώσετε το σενάριο αυτό και να υπολογίσετε τον μέσο όρο των τιμών τ , $\mu\tau$, $\kappa\tau$.

Υποδείξεις: α) Προφανώς θα πρέπει να χρησιμοποιήσετε και πάλι το module random, β) Χρήσιμη μπορεί να φανεί η μέθοδος `pop` στις λίστες η οποία επιστρέφει μία τιμή της λίστας και έπειτα την αφαιρεί από τη λίστα (αν η τιμή επαναλαμβάνεται μέσα στη λίστα τότε αφαιρείται μόνο μία φορά).