Ασκήσεις εργαστηρίου

Υπενθύμιση θεωρίας

•Δομή επιλογής

Όταν επιθυμούμε να κάνουμε κάποιον έλεγχο στα δεδομένα που χρησιμοποιούμε στον κώδικα ή στα αποτελέσματα που προκύπτουν από διάφορες πράξεις είναι απαραίτητη η χρήση μιας δομής επιλογής, δηλαδή κατάλληλων εντολών που επιτρέπουν την εκτέλεση διαφορετικών εντολών ανάλογα με το αποτέλεσμα του ελέγχου.
Η δομή επιλογής που χρησιμοποιούμε στην Python είναι το **if … elif … else**.
Εάν έχουμε μόνο έναν έλεγχο χωρίς να μας ενδιαφέρει το εναλλακτικό αποτέλεσμα μπορούμε να χρησιμοποιήσουμε απλά ένα if, δηλαδή:

if (***συνθήκη***)**:**

Η ***συνθήκη*** συνήθως σχετίζεται με τον έλεγχο μιας μεταβλητής με μία ή περισσότερες λογικές εκφράσεις.
Οι πιο βασικές λογικές εκφράσεις είναι οι εξής **>** (μεγαλύτερο), **<** (μικρότερο), **>=** (μεγαλύτερο ή ίσο), **<=** (μικρότερο ή ίσο), **==** (λογική ισότητα), **!=** (όχι ίσο, το αντίθετο της λογικής ισότητας).
Προσέξτε τη διαφορά λογικής ισότητας (δύο ίσον) με το απλό ίσον που σημαίνει ανάθεση τιμής σε μεταβλητή.
Ένας χρήσιμος έλεγχος πολλές φορές σχετίζεται με τον **έλεγχο για ζυγούς ή περιττούς αριθμούς** οπότε χρησιμοποιούμε το σύμβολο % για το υπολογισμό του υπολοίπου της διαίρεσης και έτσι ο ζυγός αριθμός είναι αυτός που έχει υπόλοιπο διαίρεσης με το 2 μηδέν δηλαδή (i%2)==0 ενώ ο περιττός (i%2)!=0.
Επίσης, για λογικές πράξεις μπορούμε να χρησιμοποιήσουμε και τα **and**, **or** και **not**.

Ασκήσεις εργαστηρίου

Υπενθύμιση θεωρίας

Το **and** συνδέει λογικές πράξεις που πρέπει να είναι όλες αληθείς για να έχει αποτέλεσμα αληθές (TRUE) δηλαδή να ισχύει και η μία και η άλλη συνθήκη **ταυτόχρονα**.
Το **or** συνδέει λογικές πράξεις που αρκεί να είναι μία αληθής για να έχει αποτέλεσμα αληθές (TRUE) δηλαδή να ισχύει είτε η μία είτε η άλλη συνθήκη.
Το **not** σχετίζεται με το αντίθετο της συνθήκης. Δηλαδή είναι αληθές (TRUE) όταν η συνθήκη δεν ισχύει.
Για παράδειγμα εάν a = 4, b =5 τότε:
Το (a>0) and (b>0) είναι αληθές ενώ το (a<0) and (b>0) όχι. Όμως το (a<0) or (b>0) είναι αληθές γιατί ισχύει η δεύτερη συνθήκη.

Εάν θέλω **περισσότερες από μία** εναλλακτικές τότε χρησιμοποιώ το:

```
if (συνθήκη):
        εντολές
else:
        εντολές
```

Εννοείται ότι οι εντολές μετά το else θα εκτελεστούν σε **οποιαδήποτε περίπτωση** δεν ισχύει η συνθήκη που υπάρχει μετά το if. Γι' αυτό και μετά το else δε χρειάζεται να γράψουμε καμία συνθήκη.

Ασκήσεις εργαστηρίου

Υπενθύμιση θεωρίας

Ενώ στη **γενική περίπτωση** χρησιμοποιώ το:

if (***συνθήκη***)**:**
    εντολές
elif (***συνθήκη***)**:**
    εντολές
elif (***συνθήκη***)**:**
    εντολές
…
else**:**
    εντολές

ΠΡΟΣΟΧΗ: στην Python μεγάλη σημασία έχει η θέση στην οποία γράφονται οι εντολές. Εάν δεν υπάρχει κάποιος συγκεκριμένος λόγος οι εντολές αρχίζουν **απαραίτητα** αριστερά.
Όμως μετά το **if, elif, else** πρέπει η επόμενη γραμμή να βρίσκεται σε **απόσταση 1 tab** από την αριστερή πλευρά της οθόνης και όλες οι γραμμές που αντιστοιχούν στην ίδια συνθήκη να είναι στοιχισμένες ακριβώς από κάτω. Αν γράψουμε και άλλο if μέσα σε υπάρχον if, elif ή else τότε η επόμενη γραμμή θα είναι ακόμα ένα tab δεξιά.
Στην Python **δε χρειάζεται** κάποια εντολή για να δηλωθεί το τέλος της δομής επιλογής. Εάν γράψουμε κάποια εντολή μετά την τελευταία συνθήκη στην αριστερή θέση, αυτή δεν ανήκει πια στη δομή επιλογής.

Ασκήσεις εργαστηρίου

Υπενθύμιση θεωρίας

•**Δομές επανάληψης**

Οι δομές επανάληψης χρησιμοποιούνται για να πραγματοποιήσουμε επαναληπτικές διαδικασίες. Διευκολύνει ιδιαίτερα την εκτέλεση μαθηματικών πράξεων όπως αθροίσματα ή γινόμενα. Οι δύο βασικές δομές επανάληψης της Python είναι η while και η for.

Η εντολή while έχει ομοιότητες με τη δομή επιλογής if γιατί απαιτεί τον έλεγχο μιας συνθήκης στην πρώτη γραμμή:

while (***συνθήκη***)**:**

Όσο ισχύει η συνθήκη επαναλαμβάνονται οι εντολές που ανήκουν στη δομή επανάληψης. Παρόμοια με το if πρέπει απαραίτητα η επόμενη γραμμή να γραφεί 1 tab πιο δεξιά. Όταν θέλουμε να γράψουμε κάποια εντολή εκτός του while τη γράφουμε κανονικά στην αριστερή θέση.
Μέσα στο while επιτρέπεται να γράφουμε και άλλες δομές επιλογής ή επανάληψης, αρκεί να έχει νόημα η εκτέλεσή τους.
ΠΡΟΣΟΧΗ: Εάν η συνθήκη στο while ισχύει πάντα χωρίς αυτό να αλλάζει στις επόμενες εντολές, τότε η εκτέλεση θα συνεχίσει για πάντα.

Ασκήσεις εργαστηρίου

Υπενθύμιση θεωρίας

•**Δομές επανάληψης**

Η εντολή for συντάσσεται ως εξής:

for i in range ( ):

Το i είναι μια ακέραια μεταβλητή που δεν χρειάζεται αρχικοποίηση προηγουμένως γιατί η τιμή της αλλάζει με βάση τις τιμές στο range. Το range μπορεί να πάρει μόνο 1 τιμή, οπότε υποδηλώνει τον αριθμό των επαναλήψεων που γίνονται π.χ. range(5), δύο τιμές, οπότε υποδηλώνει την αρχική και την τελική τιμή π.χ. range(1,5) ή τρεις τιμές οπότε υποδηλώνεται και το βήμα δηλαδή αν range(1,10,2) τότε το i θα παίρνει τιμές 1,3,5 κτλ.
Αν δεν ορίσουμε εμείς την αρχική τιμή για το i η Python θεωρεί ως πρώτη τιμή το 0.
Το βήμα μπορεί να έχει και αρνητική τιμή αλλά τότε πρέπει η αρχική τιμή να είναι μεγαλύτερη της αρχικής για να γίνουν επαναλήψεις.

Μια χρήσιμη έννοια στις δομές επανάληψης είναι η χρήση μιας μεταβλητής για την αποθήκευση των τιμών ενός αθροίσματος ή γινομένου. Η μεταβλητή αυτή πρέπει να αρχικοποιηθεί πριν την έναρξη των επαναλήψεων. Εννοείται ότι στην περίπτωση του αθροίσματος η αρχική τιμή είναι 0 ενώ στην περίπτωση του γινομένου 1 (αν η αρχική τιμή ήταν 0 θα μηδενίζονταν το γινόμενο).
Επίσης πολλές φορές μας ενδιαφέρει ο αριθμός των επαναλήψεων που έχουν πραγματοποιηθεί (π.χ. για υπολογισμό μέσου όρου) οπότε χρησιμοποιείται με την ίδια λογική μια μεταβλητή-μετρητής.

Ασκήσεις εργαστηρίου

Άσκηση 1

Γράψτε σε Python κώδικα που προσδιορίζει το είδος μιας ροής (αν είναι στρωτή ή τυρβώδης) με βάση την τιμή του αριθμού Reynolds. Για να είναι η ροή τυρβώδης πρέπει να ισχύει Re>2000, όπου $Re_x = \frac{\rho v x}{\mu}$

Άσκηση 2

Γράψτε σε Python κώδικα που να επιλύει την εξίσωση $ax^2+bx+c = 0$ για τυχαίες τιμές των a,b,c. Οι λύσεις πρέπει να παρουσιάζονται στην οθόνη και να αναφέρεται το είδος τους. Δίνεται: $\Delta = \beta^2-4\alpha\gamma$ και $x_{1,2} = \frac{-\beta \pm \sqrt{\Delta}}{2\alpha}$

Άσκηση 3

Γράψτε σε Python κώδικα που να υπολογίζει την τάση (σ) με βάση τις τιμές της δύναμης (F) και του εμβαδού της διατομής (A) ενός αντικειμένου (από τη γνωστή σχέση $\sigma = \frac{F}{A}$). Στο χρήστη θα δίνονται επιλογές ανάλογα με τη γεωμετρία της διατομής (κυκλική, τετραγωνική, ορθογώνια) και θα γίνεται έλεγχος μηδενικής τιμής. Επίσης θα ελέγχεται η τιμή της τάσης εάν είναι μεγαλύτερη από 100e6 και θα παρουσιάζεται το μήνυμα «Τάση μεγαλύτερη της αντοχής του υλικού», «Τάση μικρότερη της αντοχής του υλικού».

Ασκήσεις εργαστηρίου

Άσκηση 4

Γράψτε σε Python κώδικα που να εμφανίζει στην οθόνη τους περιττούς αριθμούς από το 1 έως το 20. Να κάνετε το ίδιο για τους αριθμούς από το 31 έως το 10 (με αντίστροφη σειρά).

Άσκηση 5

Γράψτε σε Python κώδικα που να υπολογίζει το άθροισμα των περιττών αριθμών από 1 έως Ν.

Άσκηση 6

Γράψτε σε Python κώδικα που να υπολογίζει τους Ν πρώτους όρους της ακολουθίας Fibonacci που ορίζεται ως εξής: $F_n = F_{n-1}+F_{n-2}$ για n>=2 με $F_0=0$ και $F_1=1$.

Άσκηση 7

Γράψτε σε Python κώδικα που να υπολογίζει το παραγοντικό n! = n*(n-1)*….*1.

Άσκηση 8

Υπολογίστε το άθροισμα των δυναμοσειρών: $\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$, $\sum_{n=1}^{\infty} \frac{1}{n^2}$

Λυμένες ασκήσεις

Άσκηση 1

Για να προσδιορίσουμε το είδος της ροής με βάση τον αριθμό Reynolds πρέπει να κάνουμε έλεγχο της τιμής του, εάν είναι μεγαλύτερη της τιμής 2000. Σε αυτή την περίπτωση πρέπει ο κώδικας να εμφανίσει κατάλληλο μήνυμα ότι η ροή είναι τυρβώδης (δηλαδή πιο έντονη ροή με δίνες) αλλιώς να εμφανίζει ότι η ροή είναι στρωτή (πιο ομαλή ροή).

Επειδή χρειαζόμαστε έλεγχο θα χρησιμοποιήσουμε τη δομή επιλογής if. Αφού έχουμε μόνο δύο περιπτώσεις, τη στρωτή και την τυρβώδη ροή χρειαζόμαστε μόνο δύο πιθανές επιλογές, το if και το else.

Για να γίνει ο έλεγχος όμως πρέπει αρχικά να γνωρίζουμε την τιμή του αριθμού Reynolds οπότε πρέπει να υπολογιστεί από τη σχέση Re=r*v*x/m, όπου ρ η πυκνότητα του ρευστού, v η ταχύτητά του, x κατάλληλο μήκος και μ το ιξώδες του ρευστού. Άρα αρχικά θα ορίσουμε αυτά τα τέσσερα μεγέθη, θα υπολογίσουμε τον αριθμό Reynolds και τέλος θα κάνουμε τον έλεγχο.

Με το if ελέγχουμε την περίπτωση της τυρβώδους ροής (Re>2000) ενώ στο else συμπεριλαμβάνονται όλες οι άλλες περιπτώσεις (δηλαδή Re<=2000).

Οπότε ο κώδικας γράφεται ως εξής:

Λυμένες ασκήσεις

Άσκηση 1

```
r= 1000
v= 10
x=0.2
m=0.001
Re=r*v*x/m
print(Re)
if (Re>2000):
        print("Η ροή είναι τυρβώδης")
else:
        print("Η ροή είναι στρωτή")
```

Ορισμοί βασικών μεγεθών

Υπολογισμός αριθμού Reynolds

Προαιρετικά εμφανίζουμε την τιμή του αριθμού Reynolds

Έλεγχος στρωτής-τυρβώδους ροής

Άσκηση 2

Επειδή η δευτεροβάθμια εξίσωση ανάλογα με τις τιμές των συντελεστών της μπορεί να έχει δύο πραγματικές, μία διπλή πραγματική ή μη πραγματικές (μιγαδικές) λύσεις πρέπει να χρησιμοποιήσουμε μια δομή ελέγχου για να εμφανίσουμε το είδος των λύσεων σε κάθε περίπτωση.

Γνωρίζουμε ότι το είδος των λύσεων εξαρτάται από την τιμή της διακρίνουσας Δ, δηλαδή αν η διακρίνουσα Δ έχει θετική τιμή, τότε έχουμε δύο πραγματικές λύσεις, εάν η διακρίνουσα έχει

Λυμένες ασκήσεις

Άσκηση 2

τιμή μηδέν, τότε έχουμε μία διπλή πραγματική λύση και εάν η διακρίνουσα είναι μικρότερη από μηδέν τότε έχουμε μιγαδικές λύσεις (γιατί η τετραγωνική ρίζα θα είναι αρνητική).

Οπότε το πρώτο βήμα για την κατασκευή του κώδικα είναι ο υπολογισμός της διακρίνουσας. Για να γίνει ο υπολογισμός από τη σχέση $\Delta = \beta^2-4\alpha\gamma$ πρέπει πρώτα να έχουν οριστεί τα α,β,γ.

Στη συνέχεια χρησιμοποιούμε τη δομή επιλογής στην οποία πρέπει να έχουμε τρεις περιπτώσεις, είτε ένα if, ένα elif και ένα else, είτε ένα if και δύο elif. Επιλέγουμε την πρώτη περίπτωση που είναι ισοδύναμη με τη δεύτερη γιατί εάν ελέγξουμε την περίπτωση Δ>0 και Δ=0, με το else θα έχουμε συμπεριλάβει και το Δ<0.

Τέλος, σε κάθε περίπτωση θα υπολογίζουμε τις λύσεις από τη σχέση $x_{1,2} = \frac{-\beta \pm \sqrt{\Delta}}{2\alpha}$ και θα εμφανίζουμε τα κατάλληλα μηνύματα.

Οπότε ο κώδικας γράφεται ως εξής:

```
a = 2
b = -3
c = 4
D=b**2-4*a*c
if (D>0):
    x1=(-b+D**(1/2))/(2*a)
    x2=(-b-D**(1/2))/(2*a)
```

Ορισμοί βασικών μεταβλητών

Υπολογισμός διακρίνουσας

Έλεγχος και υπολογισμός λύσεων

Λυμένες ασκήσεις

Άσκηση 2

```
        print("Υπάρχουν δύο πραγματικές ρίζες, η x1=", x1, " και η x2=",x2)
elif (D==0):
        x1=-b/(2*a)
        print("Υπάρχει μία πραγματική ρίζα, η x1=", x1)
else:
        x1=(-b+D**(1/2))/(2*a)
        x2=(-b-D**(1/2))/(2*a)
        print("Υπάρχουν δύο μιγαδικές ρίζες, η x1=", x1, " και η x2=",x2)
```

Εναλλακτικά θα μπορούσε να χρησιμοποιηθεί και το (-b+D**(1/2))/(2*a)

Η Python υποστηρίζει και μιγαδικούς αριθμούς

Άσκηση 4

Επειδή η άσκηση ζητάει να εμφανιστούν πολλοί αριθμοί χρειάζεται να χρησιμοποιηθεί δομή επανάληψης. Για να γράψουμε τους αριθμούς από το 1 έως το 20 θα χρησιμοποιήσουμε τη δομή επανάληψης for (for i in range ( )), η οποία μας διευκολύνει αρκετά.

Για να εμφανίσουμε τους περιττούς αριθμούς πρέπει να χρησιμοποιήσουμε στο range εκτός από την αρχική και την τελική τιμή δηλαδή το 1 και το 20 και το βήμα 2 για να επιλέγονται μόνο οι περιττοί αριθμοί με αρχή το 1.

Εναλλακτικά θα μπορούσαμε να χρησιμοποιήσουμε και το while π.χ. (while i<20) και έλεγχο if(i%2!=0) ώστε να εμφανίζει τους περιττούς αριθμούς σε κάθε επανάληψη, ανανεώνοντας και την τιμή του i ώστε να αυξάνεται κατά 1.

Λυμένες ασκήσεις

Άσκηση 4

Οπότε ο κώδικας γράφεται:

```
for i in range(1,20,2):
    print(i)
```

Εάν θέλουμε να εμφανίσουμε τους περιττούς από το 31 έως το 10 με αντίστροφη σειρά θα χρησιμοποιήσουμε αρνητικό βήμα -2 ως εξής:

```
for i in range(31,10,-2):
    print(i)
```

Άσκηση 5

Για να γράψουμε κώδικα που υπολογίζει άθροισμα πρέπει να χρησιμοποιήσουμε κάποια δομή επανάληψης. Επειδή το Ν θα είναι γνωστό στον κώδικα, μας διευκολύνει να χρησιμοποιήσουμε τη δομή επανάληψης for η οποία αναφέρεται σε συγκεκριμένο αριθμό επαναλήψεων.

Για τον υπολογισμό του αθροίσματος πρέπει να οριστεί μια επιπλέον μεταβλητή στην οποία θα αποθηκεύεται το άθροισμα μετά από κάθε επανάληψη. Η μεταβλητή αυτή θα αρχικοποιηθεί με την τιμή 0 πριν τη δομή επανάληψης γιατί αλλιώς ο κώδικας δε θα γνωρίζει αυτή τη μεταβλητή.

Οπότε ο κώδικας γράφεται:

Λυμένες ασκήσεις

Άσκηση 5

```
sum=0
N=5
for  i in range(1,N+1,2):
        sum=sum+i
print("Το άθροισμα των περιττών αριθμών από 1 έως", N, "είναι:", sum)
```

Αρχικοποίηση αθροίσματος (η μεταβλητή θα μπορούσε να είχε οποιαδήποτε ονομασία)

Η τελική τιμή είναι N+1 γιατί το range θα έφτανε έως την τιμή πριν το N χωρίς να το συμπεριλάβει

Στην Python μπορεί να γραφεί και: sum+=i

Άσκηση 7

Ο υπολογισμός του παραγοντικού αφορά υπολογισμό γινομένου, ο οποίος μοιάζει με τον υπολογισμό αθροίσματος με ορισμένες διαφορές.

Επειδή θέλουμε υπολογισμό πολλαπλασιασμού N αριθμών θα χρησιμοποιήσουμε δομή επανάληψης και αφού το N θα είναι γνωστό, μας διευκολύνει η χρήση της δομής for. Όμως, η αρχικοποίηση του γινομένου θα είναι με την τιμή 1 αντί για 0 όπως στο άθροισμα γιατί το 1 δεν επηρεάζει την τιμή του γινομένου ενώ το 0 θα τη μηδένιζε.

Οπότε ο κώδικας γράφεται:

Λυμένες ασκήσεις

Άσκηση 7

```
factorial=1
N=6
for  i in range(1,N+1):
        factorial=factorial*i

print("Το παραγοντικό του", N, "είναι:", factorial)
```

Αρχικοποίηση γινομένου (η μεταβλητή θα μπορούσε να είχε οποιαδήποτε ονομασία)

Η τελική τιμή είναι N+1 γιατί το range θα έφτανε έως την τιμή πριν το N χωρίς να το συμπεριλάβει

Στην Python μπορεί να γραφεί και: factorial*=i