Οικονομικό Πανεπιστήμιο Αθηνών, τμήμα Πληροφορικής
Μάθημα: Εισαγωγή στον Προγραμματισμό Υπολογιστών
Ακαδημαϊκό έτος: 2019–20
Διδάσκων: Α. Δημάκης

Τελική Εξέταση Ιανουαρίου: διάρκεια 2 ώρες

Σύνολο μονάδων: 11, Άριστα: 10

ΚΑΛΗ ΕΠΙΤΥΧΙΑ!

Επίθετο_____ΛΟΥΚ____ Όνομα:____ΣΚΑΪΓΟΥΟΚΕΡ___ ΑΜ: ____00000____

1η άσκηση. (2) Συμπληρώστε τα κενά. (Γράψτε «ΜΗΝΥΜΑ ΛΑΘΟΥΣ» εάν πρόκειται να εμφανιστεί κάποιο μήνυμα λάθους, ή «ΚΕΝΟ» αν δεν εμφανιστεί κάτι. Εάν εμφανιστούν πολλαπλές γραμμές δώστε τις στον διαθέσιμο χώρο.)

Απάντηση:

```
>>> x = '3'
>>> y = '1'
>>> x + len(y)
ΜΗΝΥΜΑ ΛΑΘΟΥΣ
>>> x = x + '-' + x
>>> x = x + '-' + x
>>> x = x + '-' + x
>>> x
'3-3-3-3-3-3-3-3'
>>> ls = 'hello'
>>> def func(ls):
        ls = 'world'
>>> ls
'hello'
>>> print(func(print(func('hello'))))
None
None
>>> from operator import add, mul
>>> add(mul(1, add(mul(2, 3), add(3, 4))), pow(2, 2))
17
>>> def f(x, y):
        def g(f, y):
            return f(x, y)
        return g
>>> f(1, 2)(lambda x, y: x + y, 3)
4
>>> ls = ['hello']
>>> print(ls[0][1])
e
>>> sum(2*x for x in {1, 2, 2})
6
>>> for x in (2*c for c in 'ba'):
        print(x)
bb
aa
```

2η άσκηση. Στην άσκηση αυτή θα δώσετε τρεις εναλλακτικές υλοποιήσεις της συνάρτησης `weave` έτσι ώστε η `weave(s)` να επιστρέφει συμβολοσειρά που αποτελείται (από αριστερά προς δεξιά) από τον 1ο χαρακτήρα της συμβολοσειράς `s` και τον μεθεπόμενό του, τον 2ο και τον μεθεπόμενο του 2ου, τον 3ο και τον μεθεπόμενο του 3ου,… κτλ.

Για παράδειγμα,

```
>>> weave('0123456789')
'0213243546576879'
>>> weave('abcdefgh')
'acbdcedfegfh'
>>> weave('egg')
'eg'
>>> weave('eg')
''
>>> weave('e')
''
```

a) **(1)** *Υλοποίηση με επαναληπτικό υπολογισμό*: Συμπληρώστε τα κενά του κώδικα που ακολουθεί χρησιμοποιώντας εντολές `while` ή `for` – χωρίς αναδρομικές κλήσεις συναρτήσεων και *comprehensions*.*

* *comprehensions είναι εκφράσεις της μορφής* `[x*x for x in range(1, 10)]`, `(x+1 for x in [1,2,3] if x % 1 == 1)`, *κτλ.*

Απάντηση:

```
def weave(s):
    """ΥΛΟΠΟΙΗΣΗ ΜΕ ΕΝΤΟΛΕΣ ΕΠΑΝΑΛΗΨΗΣ while ή for """
    """ΣΥΜΠΛΗΡΩΣΤΕ ΤΟ ΣΩΜΑ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ."""
    r = ''
    for i in range(len(s)-2):
       r += s[i] + s[i+2]
    return r
```

b) **(2)** *Υλοποίηση με επεξεργασία ακολουθίας*: Συμπληρώστε τα κενά του κώδικα που ακολουθεί χρησιμοποιώντας *comprehensions* της Python (πχ, *list ή generator comprehension*) ή/και τις ενσωματωμένες συναρτήσεις `map`, `reduce`, `filter` – χωρίς εντολές επανάληψης `while`, `for` και αναδρομικές κλήσεις συναρτήσεων.

Απάντηση:

```
def weave(s):
    """ΥΛΟΠΟΙΗΣΗ ΜΕ ΕΠΕΞΕΡΓΑΣΙΑ ΑΚΟΛΟΥΘΙΑΣ """
    """ΣΥΜΠΛΗΡΩΣΤΕ ΤΟ ΣΩΜΑ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ."""
    from operator import add
    from functools import reduce
    if len(s) < 3:
        return ''
    return reduce(add, (s[i] + s[i+2] for i in range(len(s)-2)))
```

c) **(1)** *Υλοποίηση με αναδρομή*: Συμπληρώστε τα κενά του κώδικα που ακολουθεί χρησιμοποιώντας αναδρομικές κλήσεις συναρτήσεων – χωρίς εντολές επανάληψης `while`, `for` και *comprehensions*.

Απάντηση:

```
def weave(s):
    """ΥΛΟΠΟΙΗΣΗ ΜΕ ΑΝΑΔΡΟΜΗ """
    """ΣΥΜΠΛΗΡΩΣΤΕ ΤΟ ΣΩΜΑ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ."""
    if len(s) < 3:
        return ''
    else:
        return s[0] + s[2] + weave(s[1:])
```

3η άσκηση. (2) Συμπληρώστε στο παρακάτω πλαίσιο τον ορισμό της συνάρτησης `remove_multiple` όπου η κλήση `remove_multiple(ls)` αφαιρεί τα στοιχεία της λίστας `ls` που εμφανίζονται δύο ή περισσότερες φορές και επιστρέφει `None`.

Για παράδειγμα,

```
>>> a = [2, 1, 'hello', 3, 10, 'hello', 10, 2, 10]
>>> remove_multiple(a)
>>> a
[1, 3]
```

Απάντηση:

```
"""ΔΩΣΤΕ ΤΟΝ ΟΡΙΣΜΟ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ remove_multiple."""
def remove_multiple(ls):
    multiples = [x for x in ls if ls.count(x) > 1]
    for x in multiples:
        ls.remove(x)
```

4η άσκηση. Εδώ θα υλοποιήσετε τις τάξεις `Accumulator` και `TwoWayAccumulator` των οποίων τα αντικείμενα αναπαριστούν μετρητή που η τιμή του μπορεί να αυξάνεται κατά μια οποιαδήποτε αριθμητική τιμή (θετική ή αρνητική). Τα αντικείμενα της τάξης `TwoWayAccumulator` επίσης διαθέτουν μέθοδο μείωσης τιμής και συγκρατούν τις συνολικές αυξομειώσεις που έχουν γίνει. Συμπληρώστε τα κενά του κώδικα που ακολουθεί έτσι ώστε να παράγεται το αναγραφόμενο αποτέλεσμα.

Απάντηση:

```
>>> class Accumulator:
        """(1) ΣΥΜΠΛΗΡΩΣΤΕ ΤΟΝ ΟΡΙΣΜΟ ΤΗΣ ΤΑΞΗΣ Accumulator."""
        def __init__(self, value):
            self.value = value
        def increase(self, value):
            self.value += value
        def __str__(self):
            return str(self.value)
```

```
>>> a = Accumulator(23.5)
>>> a.value
23.5
>>> print(a)
23.5
>>> a.increase(10)
>>> print(a)
33.5
>>> a.increase(100.5)
>>> print(a)
134.0
>>> a.increase(-34)
>>> print(a)
100.0
```

```
>>> # (2) ΣΥΜΠΛΗΡΩΣΤΕ ΤΟΝ ΟΡΙΣΜΟ ΤΗΣ ΤΑΞΗΣ TwoWayAccumulator:
>>> class TwoWayAccumulator(Accumulator):
        def __init__(self, value = 0):
            Accumulator.__init__(self, value)
            self.total_acc = Accumulator(0)
        def increase(self, value):
            Accumulator.increase(self, value)
            self.total_acc.increase(abs(value))
        def decrease(self, value):
            self.increase(-value)
        def total(self):
            return self.total_acc.value
```

```
>>> a2 = TwoWayAccumulator()
>>> a2.increase(10)
>>> print(a2)
10
>>> a2.decrease(10)
>>> print(a2)
0
>>> a2.total() # επιστρέφει τη συνολική αυξομείωση = 10 + 10
20
>>> a2.increase(50)
>>> print(a2)
50
>>> a2.decrease(80)
>>> print(a2)
-30
>>> a2.total() # συνολική αυξομείωση = 10 + 10 + 50 + 80
150
```