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

Κατατακτήριες εξετάσεις: διάρκεια 2 ώρες

Όλες οι ασκήσεις είναι βαθμολογικά ισοδύναμες

1η άσκηση

Συμπληρώστε στα κενά το αποτέλεσμα εκτέλεσης των παρακάτω εντολών ή τον κώδικα Python που παράγει τα αναγραφόμενα αποτελέσματα. *Συμπληρώστε «ΜΗΝΥΜΑ ΛΑΘΟΥΣ» εάν πρόκειται να εμφανιστεί κάποιο μήνυμα λάθους, ή «ΚΕΝΟ» αν δεν εμφανιστεί κάτι.*

Απάντηση:

```
>>> x = 5
>>> y, x = x + 1, y
_________________
>>> x = 10
>>> def foo():
        print(x)
>>> x += 1
>>> foo()
_________________
>>> def do():
        x = 'hello'
        foo()
>>> do()
_________________
>>> def f(x):
        def g(y, z):
            return x(z, y)
        return g
>>> h = f(pow)
>>> h(2, 3)
_________________
>>> f(lambda x, y: x + y)(2, 3)
>>> [-x for x in range(10) if x % 3 == 0]
_________________
>>> _________________
>>> a['hello']
2019
>>> def z():
        yield 2019
        yield 1
>>> for x in z():
        print(x)
_________________
```

2η άσκηση

Θεωρήστε το πρόγραμμα Python:

Απάντηση:

```
>>> def print_all_pairs(ls):
    """ΣΥΜΠΛΗΡΩΣΤΕ ΤΟ ΣΩΜΑ ΤΗΣ ΣΥΝΑΡΤΗΣΗΣ."""
```

```
>>> print_all_pairs(['a', 'b', 'c', 'd'])
a b
a c
a d
b c
b d
c d
```

Συμπληρώστε το κενό στον κώδικα της `print_all_pairs` έτσι ώστε το παραπάνω πρόγραμμα να έχει το περιγραφόμενο αποτέλεσμα, δηλαδή να εμφανίζει όλα τα δυνατά ζεύγη στοιχείων της λίστας `ls`. (Παρατηρήστε, ότι κάθε ζεύγος εμφανίζεται ακριβώς μια φορά.) *Η υλοποίησή σας θα πρέπει να δουλεύει σωστά για οποιαδήποτε λίστα συμβολοσειρών/αλφαριθμητικών `ls`, όχι μόνο για το παραπάνω παράδειγμα.*

3η άσκηση

Τι θα εμφανίσει το ακόλουθο πρόγραμμα Python όταν εκτελεστεί;

```
def what_do_i_do(s):
    if len(s) == 1:
        return print(s)
    print(s[::-1])
    what_do_i_do(s[1:])
    print(s)

what_do_i_do('hello')
```

Απάντηση:

4η άσκηση

Κατασκευάστε τη συνάρτηση `cycle(ls, i)` η οποία μετατοπίζει τα στοιχεία της λίστας `ls` κατά `i` θέσεις αριστερά. Το 1ο έως και το `i`-οστό στοιχείο ανακυκλώνονται στο τέλος της `ls`, δηλαδή μετακινούνται στις τελευταίες `i` θέσεις χωρίς όμως να αλλάξει η μεταξύ τους σειρά. *Η υλοποίησή σας θα πρέπει να δουλεύει σωστά για οποιαδήποτε λίστα `ls`, όχι μόνο για το παραπάνω παράδειγμα.*

Απάντηση:

```
>>> def cycle(ls, i):
        """ΣΥΜΠΛΗΡΩΣΤΕ ΤΟΝ ΚΩΔΙΚΑ ΩΣΤΕ ΝΑ ΕΜΦΑΝΙΖΕΤΑΙ Η
        ΠΕΡΙΓΡΑΦΟΜΕΝΗ ΕΞΟΔΟΣ."""
```

```
>>> ls = ['h', 'e', 'l', 'l', 'o']
>>> cycle(ls, 1)
>>> s
['e', 'l', 'l', 'o', 'h']
>>> cycle(ls, 1)
>>> s
['l', 'l', 'o', 'h', 'e']
>>> cycle(s, 2)
>>> s
['o', 'h', 'e', 'l', 'l']
```

5η άσκηση

Συμπληρώστε τον ορισμό της τάξης `LightBulb` στα κενά που ακολουθούν έτσι ώστε το παρακάτω πρόγραμμα να έχει το αποτέλεσμα που περιγράφεται:

```
"""ΣΥΜΠΛΗΡΩΣΤΕ ΤΟΝ ΚΩΔΙΚΑ ΩΣΤΕ ΝΑ ΕΜΦΑΝΙΖΕΤΑΙ Η ΠΕΡΙΓΡΑΦΟΜΕΝΗ ΕΞΟΔΟΣ."""

>>> class LightBulb:
```

```
light1 = LightBulb(True)
print(light1.state()) # Εμφανίζει Light is on
print(light1.state()) # Εμφανίζει Light is on
light1.press_switch()
print(light1.state()) # Εμφανίζει Light is off
light1.press_switch()
print(light1.state()) # Εμφανίζει Light is on
light1.press_switch()
print(light1.state()) # Εμφανίζει Light is off
light1.press_switch()
print(light1.state()) # Εμφανίζει Light bulb burned out
print(light1.state()) # Εμφανίζει Light bulb burned out
light1.press_switch()
print(light1.state()) # Εμφανίζει Light bulb burned out

light2 = LightBulb(False)
print(light2.state()) # Εμφανίζει Light is off
light2.press_switch()
print(light2.state()) # Εμφανίζει Light is on
light2.press_switch()
print(light2.state()) # Εμφανίζει Light is off
light2.press_switch()
print(light2.state()) # Εμφανίζει Light is on
light2.press_switch()
print(light2.state()) # Εμφανίζει Light bulb burned out
```