Εργαστήριο 9

Εισαγωγή στον Προγραμματισμό Υπολογιστών

Περιεχόμενα

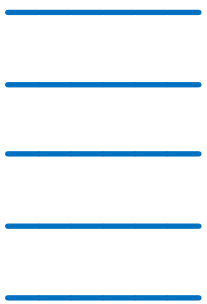
- Iterators & Generators
  - `reversed`
    - Υλοποίηση με *generator* `my_reversed`
  - `zip`
    - Υλοποίηση με *generator* `my_zip`
    - Εσωτερικό γινόμενο διανυσμάτων
  - `enumerate`
  - Ακολουθίες πρώτων αριθμών

Ενσωματωμένος iterator `reversed`

- Ορισμένες συναρτήσεις της Python επιστρέφουν *iterators*:

`reversed(seq)`: σαρώνει ανάποδα τα στοιχεία της ακολουθίας `seq`

```
>>> for c in reversed('Hello'):
        print(c)
```



Ενσωματωμένος iterator `reversed`

- Ορισμένες συναρτήσεις της Python επιστρέφουν *iterators*:

`reversed(seq)`: σαρώνει ανάποδα τα στοιχεία της ακολουθίας `seq`

```
>>> for c in reversed('Hello'):
        print(c)
o
l
l
e
H
```

Ενσωματωμένος iterator `reversed`: υλοποίηση με *generator*

- **Generators** (γεννήτριες): iterators που όταν κληθεί η `next`, επιστρέφουν το επόμενο στοιχείο σύμφωνα με υπολογισμό που δίνεται από τον προγραμματιστή

- Υλοποίηση `my_reversed`:

```
>>> def my_reversed(seq):
        for i in range(__________,__________,__________):
            yield seq[i]

>>> for c in my_reversed('Hello'):
        print(c)
o
l
l
e
H
```

Ενσωματωμένος iterator `reversed`: υλοποίηση με *generator*

- **Generators** (γεννήτριες): iterators που όταν κληθεί η `next`, επιστρέφουν το επόμενο στοιχείο σύμφωνα με υπολογισμό που δίνεται από τον προγραμματιστή

- Υλοποίηση `my_reversed`:

```
>>> def my_reversed(seq):
        for i in range(len(seq)-1, -1, -1):
            yield seq[i]

>>> for c in my_reversed('Hello'):
        print(c)
o
l
l
e
H
```

Ενσωματωμένος iterator `reversed`: υλοποίηση με *generator*

- **Generators** (γεννήτριες): iterators που όταν κληθεί η `next`, επιστρέφουν το επόμενο στοιχείο σύμφωνα με υπολογισμό που δίνεται από τον προγραμματιστή

- Ισοδύναμη υλοποίηση `my_reversed` με *generator comprehension*:

```
>>> def my_reversed(seq):
        return (______ for i in ____________________________)


>>> for c in my_reversed('Hello'):
        print(c)
o
l
l
e
H
```

Ενσωματωμένος iterator `reversed`: υλοποίηση με *generator*

- **Generators** (γεννήτριες): iterators που όταν κληθεί η `next`, επιστρέφουν το επόμενο στοιχείο σύμφωνα με υπολογισμό που δίνεται από τον προγραμματιστή

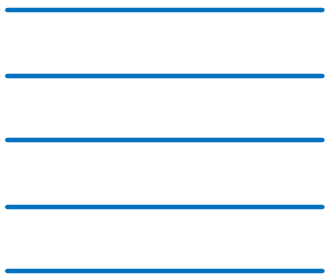
- Ισοδύναμη υλοποίηση `my_reversed` με *generator comprehension*:

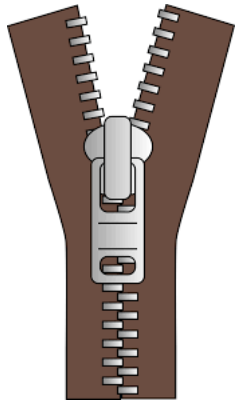
```
>>> def my_reversed(seq):
        return (seq[i] for i in range(len(seq)-1, -1, -1))


>>> for c in my_reversed('Hello'):
        print(c)
o
l
l
e
H
```

Ενσωματωμένος iterator `zip`

- `zip(iterable1, iterable2)`: επιστρέφει *iterator* που δίνει ζεύγη στοιχείων από το κάθε iterable

```
>>> for x in zip('Hello','world'):
        print(x)
```

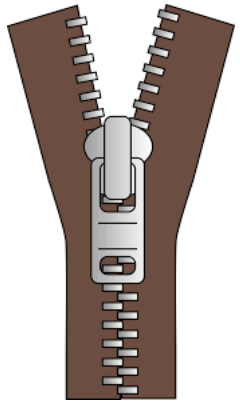




Ενσωματωμένος iterator `zip`

- `zip(iterable1, iterable2)`: επιστρέφει *iterator* που δίνει ζεύγη στοιχείων από το κάθε iterable

```
>>> for x in zip('Hello','world'):
        print(x)

('H', 'w')
('e', 'o')
('l', 'r')
('l', 'l')
('o', 'd')
```



Ενσωματωμένος iterator `zip`: υλοποίηση με *generator*

- Υλοποίηση `zip` με *generator*:

```
>>> def my_zip(s, t):
        for i in range(_____________________):
            yield _________________

>>> for x in my_zip('Hello','world'):
        print(x)

('H', 'w')
('e', 'o')
('l', 'r')
('l', 'l')
('o', 'd')
```

Ενσωματωμένος iterator `zip`: υλοποίηση με *generator*

- Υλοποίηση `zip` με *generator*:

```
>>> def my_zip(s, t):
        for i in range(min(len(s), len(t))):
            yield (s[i], t[i])

>>> for x in my_zip('Hello','world'):
        print(x)

('H', 'w')
('e', 'o')
('l', 'r')
('l', 'l')
('o', 'd')
```

Εσωτερικό γινόμενο διανυσμάτων

- Υπολογισμός εσωτερικού γινομένου διανυσμάτων:

```
>>> def inner_prod(x, y):
        return sum(_____ for _____ in _________)

>>> inner_prod((1, 3, 9), (2, 2, -1))
-1
```

Εσωτερικό γινόμενο διανυσμάτων

- Υπολογισμός εσωτερικού γινομένου διανυσμάτων:

```
>>> def inner_prod(x, y):
        return sum(xi*yi for xi, yi in zip(x, y))

>>> inner_prod((1, 3, 9), (2, 2, -1))
-1
```

Ενσωματωμένος iterator `enumerate`: εναλλακτική υλοποίηση

- `enumerate(iterable)`:

```
>>> for x in enumerate('Hello'):
        print(x)
(0, 'H')
(1, 'e')
(2, 'l')
(3, 'l')
(4, 'o’)
```

- Ισοδύναμα με `zip`:

```
>>> for x in _____________________________________:
        print(x)
(0, 'H')
(1, 'e')
(2, 'l')
(3, 'l')
(4, 'o')
```

Ενσωματωμένος iterator `enumerate`: εναλλακτική υλοποίηση

- `enumerate(iterable)`:

```
>>> for x in enumerate('Hello'):
        print(x)
(0, 'H')
(1, 'e')
(2, 'l')
(3, 'l')
(4, 'o’)
```

- Ισοδύναμα με `zip`:

```
>>> for x in zip(range(len('Hello')), 'Hello'):
        print(x)
(0, 'H')
(1, 'e')
(2, 'l')
(3, 'l')
(4, 'o')
```

Ακολουθία πρώτων αριθμών

- Σάρωση όλων των πρώτων αριθμών :

```
>>> def primes():
        from primes import isprime
        current = 2
        while True:
            _____________________________
            _____________________________
            _____________________________

>>> iterator = primes()
>>> next(iterator)
2
>>> next(iterator)
3
>>> next(iterator)
5
```

Ακολουθία πρώτων αριθμών

- Σάρωση όλων των πρώτων αριθμών:

```
>>> def primes():
        from primes import isprime
        current = 2
        while True:
            if isprime(current):
                yield current
            current += 1

>>> iterator = primes()
>>> next(iterator)
2
>>> next(iterator)
3
>>> next(iterator)
5
```

Ακολουθία πρώτων αριθμών

- Εμφάνιση αρχικών 1000 πρώτων:

```
>>> for i, n in zip(____________, primes()):
        print(n)
```

Ακολουθία πρώτων αριθμών

- Εμφάνιση αρχικών 1000 πρώτων:

```
>>> for i, n in zip(range(1000), primes()):
        print(n)
```

  - Ισοδύναμα, εφόσον δε χρησιμοποιούμε το όνομα `i`:

```
>>> for _, n in zip(range(1000), primes()):
        print(n)
```