

# Εργαστήριο 6

Εισαγωγή στον Προγραμματισμό Υπολογιστών

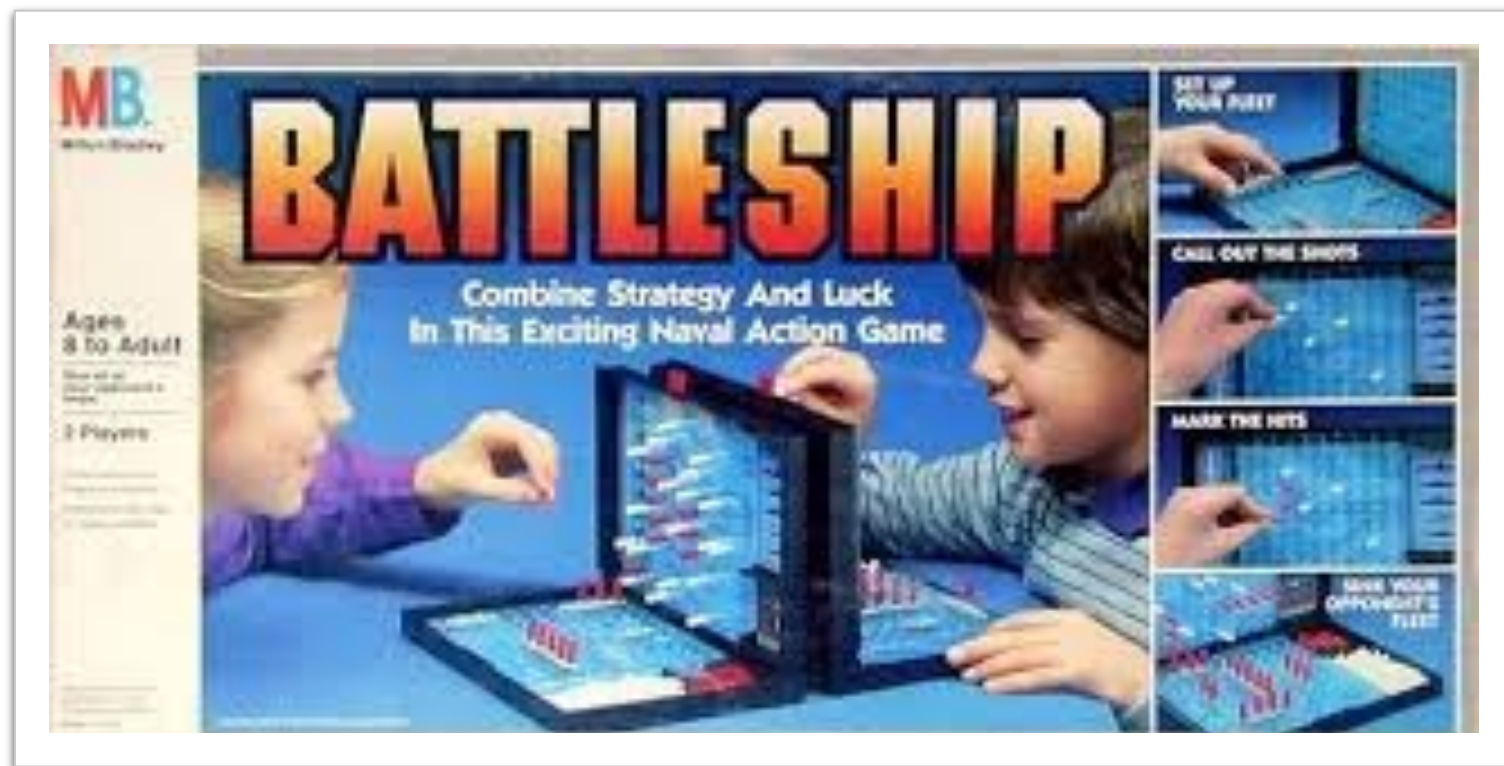
# Περιεχόμενα

## 1. Αφαίρεση δεδομένων

- *Ναυμαχία*

## 2. Ακολουθίες

- Λίστα
- `range`
- Εντολή `for`
- *List comprehensions*



# Παράδειγμα: *Ναυμαχία*

[https://en.wikipedia.org/wiki/Battleship\\_\(game\)\)](https://en.wikipedia.org/wiki/Battleship_(game)))



# Παράδειγμα: Ναυμαχία

- Κατεβάστε το module [position.py](#) από eclass -> Έγγραφα -> Εργαστήρια -> 6
- Οι συναρτήσεις του `position` αναπαριστούν το σύνθετο δεδομένο "τοποθεσία"
  - **Κατασκευαστής:** `p = pos(x, y)` επιστρέφει την τοποθεσία `p` με συντεταγμένες `x, y`
    - `x, y` είναι ακέραιοι αριθμοί
  - **Επιλογείς (selectors):**
    - `get_x(p)`: επιστρέφει την οριζόντια συντεταγμένη (αριθμό στήλης) της θέσης `p`
    - `get_y(p)`: επιστρέφει την κάθετη συντεταγμένη (αριθμό γραμμής) της θέσης `p`

# Παράδειγμα: Ναυμαχία

1. Συμπληρώστε τα κενά με text editor και αποθηκεύστε τις αλλαγές

Αρχείο position.py:

```
def pos(x, y):  
    return [x, y]  
  
def get_x(p):  
    return _____  
  
def get_y(p):  
    return _____
```

2. Ελέγξτε από διαδραστικό περιβάλλον

• Παράδειγμα κλήσεων

```
>>> from position import *  
>>> p = pos(3, 9)  
>>> get_x(p)  
3  
>>> get_y(p)  
9
```

	1	2	3	4	5	6	7	8	9	10
1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1	10,1
2	1,2	2,2	3,2	4,2	5,2	6,2	7,2	8,2	9,2	10,2
3	1,3	2,3	3,3	4,3	5,3	6,3	7,3	8,3	9,3	10,2
4	1,4	2,4	3,4	4,4	5,4	6,4	7,4	8,4	9,4	10,4
5	1,5	2,5	3,5	4,5	5,5	6,5	7,5	8,5	9,5	10,5
6	1,6	2,6	3,6	4,6	5,6	6,6	7,6	8,6	9,6	10,6
7	1,7	2,7	3,7	4,7	5,7	6,7	7,7	8,7	9,7	10,7
8	1,8	2,8	3,8	4,8	5,8	6,8	7,8	8,8	9,8	10,8
9	1,9	2,9	3,9	4,9	5,9	6,9	7,9	8,9	9,9	10,9
10	1,10	2,10	3,10	4,10	5,10	6,10	7,10	8,10	9,10	10,10

# Παράδειγμα: Ναυμαχία

1. Συμπληρώστε τα κενά με text editor και αποθηκεύστε τις αλλαγές

Αρχείο position.py:

```
def pos(x, y):  
    return [x, y]  
  
def get_x(p):  
    return p[0]  
  
def get_y(p):  
    return p[1]
```

2. Ελέγξτε από διαδραστικό περιβάλλον

• Παράδειγμα κλήσεων

```
>>> from position import *  
>>> p = pos(3, 9)  
>>> get_x(p)  
3  
>>> get_y(p)  
9
```

	1	2	3	4	5	6	7	8	9	10
1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1	10,1
2	1,2	2,2	3,2	4,2	5,2	6,2	7,2	8,2	9,2	10,2
3	1,3	2,3	3,3	4,3	5,3	6,3	7,3	8,3	9,3	10,2
4	1,4	2,4	3,4	4,4	5,4	6,4	7,4	8,4	9,4	10,4
5	1,5	2,5	3,5	4,5	5,5	6,5	7,5	8,5	9,5	10,5
6	1,6	2,6	3,6	4,6	5,6	6,6	7,6	8,6	9,6	10,6
7	1,7	2,7	3,7	4,7	5,7	6,7	7,7	8,7	9,7	10,7
8	1,8	2,8	3,8	4,8	5,8	6,8	7,8	8,8	9,8	10,8
9	1,9	2,9	3,9	4,9	5,9	6,9	7,9	8,9	9,9	10,9
10	1,10	2,10	3,10	4,10	5,10	6,10	7,10	8,10	9,10	10,10

# Παράδειγμα: Ναυμαχία

Αρχείο [battleship.py](#):

```
def ship(p1,p2):  
    """Constructs a ship instance.  
    p1 -- bow position (front)  
    p2 -- stern position (rear)  
    """  
    return [p1, p2]  
  
def ship_pos(s, p):  
    """Checks if ship occupies position.  
    s -- ship  
    p -- position  
    """
```

- Παράδειγμα χρήσης

```
>>> bow = pos(3,2)  
>>> stern = pos(6,2)  
>>> s = ship(bow, stern)  
>>> s  
[[3, 2], [6, 2]]  
>>> ship_pos(s, pos(4, 2))  
True  
>>> ship_pos(s, pos(7, 2))  
False
```

Αποθηκεύστε  
και ελέγξτε



# Παράδειγμα: Ναυμαχία

Αρχείο [battleship.py](#):

```
def ship(p1,p2):
    """Constructs a ship instance.
    p1 -- bow position (front)
    p2 -- stern position (rear)
    """
    return [p1, p2]

def ship_pos(s, p):
    """Checks if ship occupies position.
    s -- ship
    p -- position
    """
    if get_x(s[0]) == get_x(s[1]):
        return get_x(s[0]) == get_x(p) and (get_y(s[0]) <= get_y(p) <= get_y(s[1]) \
            or get_y(s[1]) <= get_y(p) <= get_y(s[0]))
    else:
        return get_y(s[0]) == get_y(p) and (get_x(s[0]) <= get_x(p) <= get_x(s[1]) \
            or get_x(s[1]) <= get_x(p) <= get_x(s[0]))
```

- Παράδειγμα χρήσης

```
>>> bow = pos(3,2)
>>> stern = pos(6,2)
>>> s = ship(bow, stern)
>>> s
[[3, 2], [6, 2]]
>>> ship_pos(s, pos(4, 2))
True
>>> ship_pos(s, pos(7, 2))
False
```

Αποθηκεύστε  
και ελέγξτε

# Παράδειγμα: Ναυμαχία


- Χειρισμός πίνακα παιχνιδιού
  - `board(ship_list)`: κατασκευάζεται αναπαράσταση του πίνακα παιχνιδιού όπου τα πλοία στη λίστα `ship_list` έχουν τοποθετηθεί (χωρίς να εμφανιστούν)
  - `is_occupied(board, pos)`: επιστρέφει `True` εάν η τοποθεσία `pos` καταλαμβάνεται από πλοίο
- Τρόπος αναπαράστασης πίνακα παιχνιδιού: συνάρτηση `func(pos)` όπου επιστρέφει `True` εάν υπάρχει πλοίο στην τοποθεσία `pos`

# Παράδειγμα: Ναυμαχία

Αρχείο battleship.py:

```
def board(ship_list):  
    """Constructs a battleship game board.  
    ship_list -- list of ships to attach on the board  
    """  
    def board_func(pos):  
        i = 0  
        while i < len(ship_list):  
            if ship_pos(ship_list[i], pos):  
                _____  
                i += 1  
        return _____  
    return board_func  
  
def is_occupied(board_func, pos):  
    """Checks if board position is occupied by a ship.  
    board -- game board  
    pos -- position to check  
    Return value if True if position in pos is occupied by a ship on the board.  
    """  
    return _____(_____)
```

Αποθηκεύστε  
και ελέγξτε



- **Παράδειγμα χρήσης**


```
>>> ship1 = ship(pos(1,1), pos(3,1))  
>>> ship2 = ship(pos(4,3), pos(4,6))  
>>> ships = [ship1, ship2]  
>>> b = board(ships)  
>>> is_occupied(b, pos(2,1))  
True  
>>> is_occupied(b, pos(4,2))  
False
```

# Παράδειγμα: Ναυμαχία

Αρχείο battleship.py:

```
def board(ship_list):  
    """Constructs a battleship game board.  
    ship_list -- list of ships to attach on the board  
    """  
    def board_func(pos):  
        i = 0  
        while i < len(ship_list):  
            if ship_pos(ship_list[i], pos):  
                return True  
            i += 1  
        return False  
    return board_func  
  
def is_occupied(board_func, pos):  
    """Checks if board position is occupied by a ship.  
    board -- game board  
    pos -- position to check  
    Return value if True if position in pos is occupied by a ship on the board.  
    """  
    return board_func(pos)
```

Αποθηκεύστε  
και ελέγξτε



- Παράδειγμα χρήσης

```
>>> ship1 = ship(pos(1,1), pos(3,1))  
>>> ship2 = ship(pos(4,3), pos(4,6))  
>>> ships = [ship1, ship2]  
>>> b = board(ships)  
>>> is_occupied(b, pos(2,1))  
True  
>>> is_occupied(b, pos(4,2))  
False
```

# Παράδειγμα: Ναυμαχία

- Συμπληρώστε τα κενά ώστε το πρόγραμμα σας να αναπαριστά τον πίνακα παιχνιδιού αριστερά

	1	2	3	4	5	6	7	8	9	10
1	■	■	■	■						
2								■		
3								■		
4								■		
5								■		
6								■		
7										
8										
9						■	■	■		
10										

- Παράδειγμα χρήσης

```
>>> ship1 = ship(pos(1,1), pos(4,1))
>>> ship2 = _____
>>> ship3 = _____
>>> ships = [_____, _____, _____]
>>> game_board = board(_____)
>>> is_occupied(game_board, pos(2,1))
True
>>> is_occupied(game_board, pos(8,7))
False
```

# Παράδειγμα: Ναυμαχία

- Συμπληρώστε τα κενά ώστε το πρόγραμμα σας να αναπαριστά τον πίνακα παιχνιδιού αριστερά

	1	2	3	4	5	6	7	8	9	10
1	■	■	■	■						
2								■		
3								■		
4								■		
5								■		
6								■		
7										
8										
9						■	■	■		
10										

- Παράδειγμα χρήσης

```
>>> ship1 = ship(pos(1,1), pos(4,1))
>>> ship2 = ship(pos(8,2), pos(8,6))
>>> ship3 = ship(pos(6,9), pos(8,9))
>>> ships = [ship1, ship2, ship3]
>>> game_board = board(ships)
>>> is_occupied(game_board, pos(2,1))
True
>>> is_occupied(game_board, pos(8,7))
False
```

# Παράδειγμα: Ναυμαχία

Αποθηκεύστε και ελέγξτε

Αρχείο battleship.py:

```
def print_board(board):  
    """Print battleship game board.  
  
    board -- game board to print  
  
    Returns None.  
    """  
    """ΣΥΜΠΛΗΡΩΣΤΕ ΤΟΝ ΚΩΔΙΚΑ ΣΑΣ ΕΔΩ"""
```

- Παράδειγμα χρήσης

```
>>> ship1 = ship(pos(1,1), pos(4,1))  
>>> ship2 = ship(pos(8,2), pos(8,6))  
>>> ship3 = ship(pos(6,9), pos(8,9))  
>>> ships = [ship1, ship2, ship3]  
>>> game_board = board(ships)  
>>> print_board(game_board)  
1111000000  
0000000100  
0000000100  
0000000100  
0000000100  
0000000100  
0000000100  
0000000000  
0000000000  
0000011100  
0000000000
```





# Παράδειγμα: Ναυμαχία

- Γράψαμε το πρόγραμμα με τέτοιο τρόπο ώστε εάν γίνουν αλλαγές σε κάποια τμήματα να μην είναι απαραίτητο να πρέπει να αλλαχθεί ο υπόλοιπος κώδικας
- Για παράδειγμα, θα αλλάξουμε την αναπαράσταση των τοποθεσιών

# Παράδειγμα: Ναυμαχία

- Αλλάξτε την αναπαράσταση της τοποθεσίας στο module `position.py` με έναν αριθμό από 1 έως 100

```
def pos(x, y):  
    return _____
```

```
def get_x(p):  
    return _____
```

```
def get_y(p):  
    return _____
```

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	11	12	13	14	15	16	17	18	19	20
3	21	22	23	24	25	26	27	28	29	30
4	31	32	33	34	34	36	37	38	39	40
5	41	42	43	44	45	46	47	48	49	50
6	51	52	53	54	55	56	57	58	59	60
7	61	62	63	64	65	66	67	68	69	70
8	71	72	73	74	75	76	77	78	79	80
9	81	82	83	84	85	86	87	88	89	90
10	91	92	93	94	95	96	97	98	99	100

# Παράδειγμα: Ναυμαχία

- Αλλάξτε την αναπαράσταση της τοποθεσίας στο `module position.py` με έναν αριθμό από 1 έως 100

```
def pos(x, y):  
    return x+10*(y-1)  
  
def get_x(p):  
    return (p-1) % 10 + 1  
  
def get_y(p):  
    return (p-1) // 10 + 1
```

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	11	12	13	14	15	16	17	18	19	20
3	21	22	23	24	25	26	27	28	29	30
4	31	32	33	34	34	36	37	38	39	40
5	41	42	43	44	45	46	47	48	49	50
6	51	52	53	54	55	56	57	58	59	60
7	61	62	63	64	65	66	67	68	69	70
8	71	72	73	74	75	76	77	78	79	80
9	81	82	83	84	85	86	87	88	89	90
10	91	92	93	94	95	96	97	98	99	100

# Παράδειγμα: Ναυμαχία

- Το παράδειγμα θα πρέπει να δουλεύει ακόμα και μετά την αλλαγή της αναπαράστασης
- Ο κώδικας των `ship`, `ship_pos`, `board`, `print_board`, `is_occupied` δεν χρειάζεται να αλλάξει

- Παράδειγμα χρήσης(\*):

```
>>> ship1 = ship(pos(1,1), pos(4,1))
>>> ship2 = ship(pos(8,2), pos(8,6))
>>> ship3 = ship(pos(6,9), pos(8,9))
>>> ships = [ship1, ship2, ship3]
>>> game_board = board(ships)
>>> print_board(game_board)
1111000000
0000000100
0000000100
0000000100
0000000100
0000000100
0000000000
0000000000
0000011100
0000000000
```

(\* ) Για να χρησιμοποιηθεί το `position.py` μετά τις αλλαγές, ίσως χρειαστεί να εκτελέσετε τις εντολές:

```
>>> import sys
>>> del sys.modules['position']
>>> from position import *
```

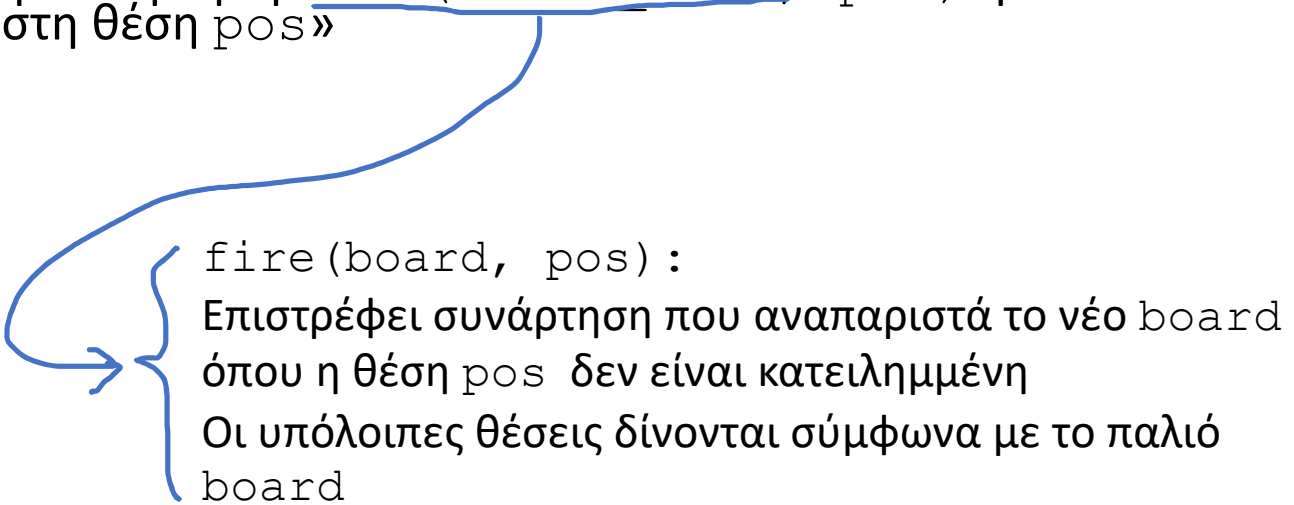
# Παράδειγμα: Ναυμαχία

Επεκτάσεις

# Παράδειγμα: Ναυμαχία

- Στο `battleship.py` προσθέστε επίσης τη συνάρτηση `fire(board_func, pos)` η οποία «βυθίζει το μέρος το πλοίου που βρίσκεται στη θέση `pos`»

```
def fire(board_func, pos):  
    """Fire.  
    board_func -- game board  
    pos -- position to fire  
    """  
  
    def new_board_func(p):  
        if p == pos:  
            return False  
        else:  
            return board_func(p)  
    return new_board_func
```



`fire(board, pos):`  
Επιστρέφει συνάρτηση που αναπαριστά το νέο `board`  
όπου η θέση `pos` δεν είναι κατειλημμένη  
Οι υπόλοιπες θέσεις δίνονται σύμφωνα με το παλιό  
`board`

# Παράδειγμα: Ναυμαχία

- Στο `battleship.py` προσθέστε επίσης τη συνάρτηση `game_over(board)` η οποία ελέγχει εάν έχει τελειώσει το παιχνίδι

`game_over(board)` :  
Επιστρέφει True εάν καμία θέση του board δεν είναι κατειλημμένη

```
def game_over(board):  
    """Checks if all ships are sunk."""  
    y = 1  
    while y <= 10:  
        x = 0  
        while x <= 10:  
            if is_occupied(board, pos(x, y)):  
                return False  
            x += 1  
        y += 1  
    return True
```

# Παράδειγμα: Ναυμαχία

- Ολοκλήρωση του παιχνιδιού:
  - Επαναληπτικά ο χρήστης δίνει συντεταγμένες του στόχου
  - Εάν ο στόχος βρίσκεται τμήμα πλοίου, το τμήμα αυτό βυθίζεται
  - Το παιχνίδι τελειώνει όταν βυθιστούν όλα τα τμήματα πλοίων
- Εκτέλεση:
  - Κατεβάστε το [game\\_loop.py](#) το οποίο υλοποιεί τα παραπάνω από το eclass

```
>>> from game_loop import *
```
- Οδηγίες:
  - *Παίζεται από δύο παίκτες, όπου ο καθένας εκτελεί το δικό του πρόγραμμα τοποθετώντας τα πλοία όπως θέλει*
  - *Όταν παίζει ένας παίκτης, λέει στον άλλον ποιο στόχο να εισάγει και ο άλλος απαντά εάν χτύπησε ή όχι κάποιο πλοίο*
- Επεκτάση:
  - Κάθε παίκτης μπορεί να εκτελεί εκτός από το παραπάνω πρόγραμμα, ένα άλλο πρόγραμμα σε διπλανό παράθυρο όπου θα εισάγει και εμφανίζει τις επιτυχίες και τις αποτυχίες των βολών του



Λίστα

# Λίστα

- Κατασκευή λίστας:

```
>>> basket = ['apple', 'peach', 'banana', 'orange', 'orange', \
               'apple', 'orange', 'banana']
```

- Εισαγωγή επιπλέον στοιχείων στο τέλος: πχ, 'apple' και 'pear'

```
>>> basket = basket + _____
```

# Λίστα

- Κατασκευή λίστας:

```
>>> basket = ['apple', 'peach', 'banana', 'orange', 'orange', \
              'apple', 'orange', 'banana']
```

- Εισαγωγή επιπλέον στοιχείων στο τέλος: πχ, 'apple' και 'pear'

```
>>> basket = basket + ['apple', 'pear']
```

- Εμφάνιση όλων των στοιχείων με εντολή while:

```
>>> i = ____
>>> while i <= len(basket)-1:
    print(____)
    _____
```

# Λίστα

- Κατασκευή λίστας:

```
>>> basket = ['apple', 'peach', 'banana', 'orange', 'orange', \
               'apple', 'orange', 'banana']
```

- Εισαγωγή επιπλέον στοιχείων στο τέλος: πχ, 'apple' και 'pear'

```
>>> basket = basket + ['apple', 'pear']
```

- Εμφάνιση όλων των στοιχείων με εντολή `while`:

```
>>> i = 0
>>> while i <= len(basket)-1:
    print(basket[i])
    i = i + 1
```

- Εμφάνιση στοιχείων με εντολή `for`:

```
>>> for _____ in _____:
    print(_____)
```

# Λίστα

- Κατασκευή λίστας:

```
>>> basket = ['apple', 'peach', 'banana', 'orange', 'orange', \
              'apple', 'orange', 'banana']
```

- Εισαγωγή επιπλέον στοιχείων στο τέλος: πχ, 'apple' και 'pear'

```
>>> basket = basket + ['apple', 'pear']
```

- Εμφάνιση όλων των στοιχείων με εντολή `while`:

```
>>> i = 0
>>> while i <= len(basket)-1:
    print(basket[i])
    i = i + 1
```

- Εμφάνιση στοιχείων με εντολή `for`:

```
>>> for fruit in basket:
    print(fruit)
```

# Λίστα

- Εμφάνιση στοιχείων με *list comprehension*

```
>>> [_____ for fruit in basket]
```

```
apple
```

```
peach
```

```
banana
```

```
orange
```

```
orange
```

```
apple
```

```
orange
```

```
banana
```

```
[None, None, None, None, None, None, None, None, None, None]
```

- Σημείωση: η λίστα που κατασκευάζεται δεν εμφανίζεται όταν η έκφραση αποτιμηθεί εκτός του διαδραστικού περιβάλλοντος

# Λίστα

- Εμφάνιση στοιχείων με *list comprehension*

```
>>> [print(fruit) for fruit in basket]
```

```
apple
```

```
peach
```

```
banana
```

```
orange
```

```
orange
```

```
apple
```

```
orange
```

```
banana
```

```
[None, None, None, None, None, None, None, None, None, None]
```

- Σημείωση: η λίστα που κατασκευάζεται δεν εμφανίζεται όταν η έκφραση αποτιμηθεί εκτός του διαδραστικού περιβάλλοντος

# Λίστα

- Συνάρτηση που μετράει πόσα φρούτα ενός συγκεκριμένου τύπου περιέχονται στο καλάθι:

```
>>> def count_fruit(basket, fruit):  
    """Number of fruits of a certain type contained in basket.  
    basket - a list of fruits  
    fruit - fruit type to count  
    """  
  
    _____  
  
    _____  
  
    _____  
  
    _____  
  
    return _____
```

```
>>> count_fruit(basket, 'apple')
```

```
3
```



# Λίστα

- Συνάρτηση που μετράει πόσα φρούτα ενός συγκεκριμένου τύπου περιέχονται στο καλάθι:

```
>>> def count_fruit(basket, fruit):  
    """Number of fruits of a certain type contained in basket.  
    basket - a list of fruits  
    fruit - fruit type to count  
    """  
  
    count = 0  
    for item in basket:  
        if item == fruit:  
            count += 1  
    return count
```

```
>>> count_fruit(basket, 'apple')
```

```
3
```

# Λίστα

- Υλοποίηση με επεξεργασία ακολουθίας (list comprehension, συγκέντρωση)

```
>>> def count_fruit(basket, fruit):  
    """Number of fruits of a certain type contained in basket.  
    basket - a list of fruits  
    fruit - fruit type to count  
    """  
    return sum([1 for _ in basket if _ == fruit])
```

```
>>> count_fruit(basket, 'apple')
```

```
3
```

# Λίστα

- Υλοποίηση με επεξεργασία ακολουθίας (list comprehension, συγκέντρωση)

```
>>> def count_fruit(basket, fruit):  
    """Number of fruits of a certain type contained in basket.  
    basket - a list of fruits  
    fruit - fruit type to count  
    """  
    return len([x for x in basket if x == fruit])
```

```
>>> count_fruit(basket, 'apple')
```

```
3
```

# Επεξεργασία ακολουθιών

- Εύρεση φρούτου που περιέχεται περισσότερες φορές στο καλάθι:

```
>>> basket
['apple', 'peach', 'banana', 'orange', 'orange', 'apple', 'orange', 'banana']
>>> fruit_count = [_____ for fruit in basket]
>>> fruit_count
[['apple', 2], ['peach', 1], ['banana', 2], ['orange', 3], ['orange', 3],
 ['apple', 2], ['orange', 3], ['banana', 2]]
>>> from functools import reduce
>>> max_fruit = reduce(lambda x, y: _____, fruit_count)
>>> max_fruit
['orange', 3]
>>> _____[0]
'orange'
```

# Επεξεργασία ακολουθιών

- Εύρεση φρούτου που περιέχεται περισσότερες φορές στο καλάθι:

```
>>> basket
['apple', 'peach', 'banana', 'orange', 'orange', 'apple', 'orange', 'banana']
>>> fruit_count = [[x, count_fruit(basket, x)] for fruit in basket]
>>> fruit_count
[['apple', 2], ['peach', 1], ['banana', 2], ['orange', 3], ['orange', 3],
 ['apple', 2], ['orange', 3], ['banana', 2]]
>>> from functools import reduce
>>> max_fruit = reduce(lambda x, y: x if x[1] > y[1] else y, fruit_count)
>>> max_fruit
['orange', 3]
>>> max_fruit[0]
'orange'
```

# Range

- Η range είναι χρήσιμη όταν χρειαζόμαστε δείκτες/απαρίθμηση

```
>>> len(basket)
10
>>> index_range = range(len(basket))
>>> list(index_range)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> for i in index_range:
    print(str(i)+' : '+'_____')
1: apple
2: peach
3: banana
4: orange
5: orange
6: apple
7: orange
8: banana
9: apple
10: pear
```

# Range

- Η range είναι χρήσιμη όταν χρειαζόμαστε δείκτες/απαρίθμηση

```
>>> len(basket)
10
>>> index_range = range(len(basket))
>>> list(index_range)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> for i in index_range:
    print(str(i+1)+' : '+ basket[i])
1: apple
2: peach
3: banana
4: orange
5: orange
6: apple
7: orange
8: banana
9: apple
10: pear
```