

ENG 346

Data Structures and Algorithms for Artificial Intelligence

Lists Tuples Maps

Dr. Mehmet PEKMEZCİ

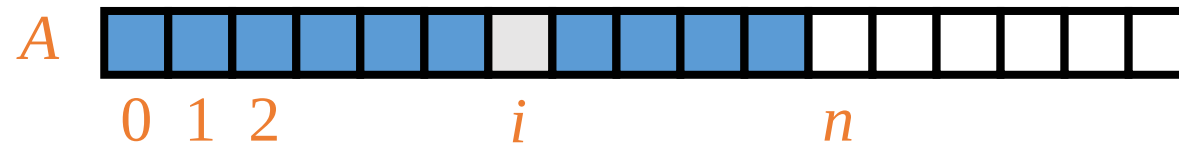
mpekmezci@gtu.edu.tr

<https://github.com/mehmetpekmezci/GTU-ENG-346>

ENG-346-FALL-2025 Teams code is **0uv7jlm**

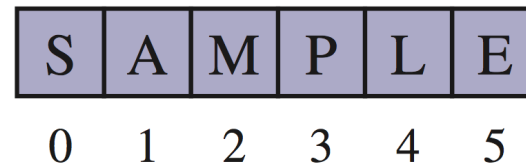
Sequences

- Built-in types: list, tuple, and str
- Sequence supports indexing, e.g. $A[i]$
- These types use an *array* to represent the sequence.
 - An array is a set of memory locations that can be addressed using consecutive indices
 - Indices start with 0.

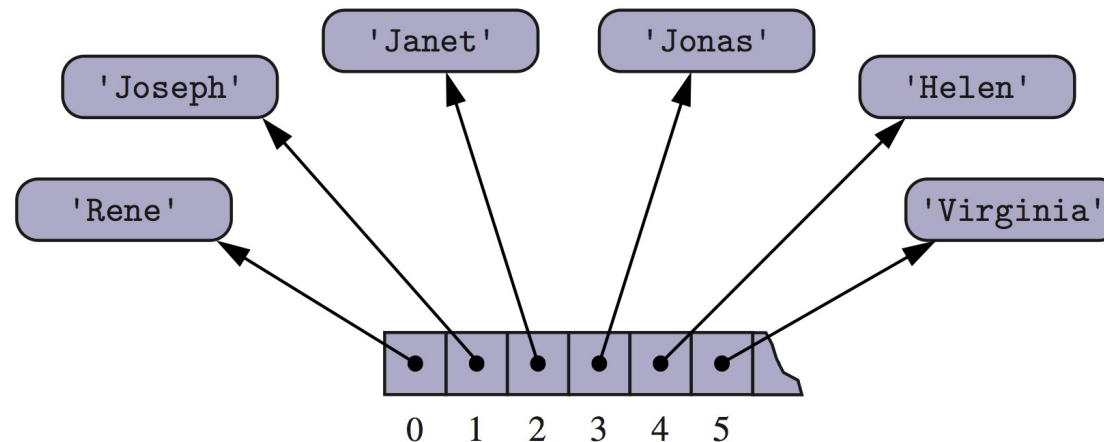


Arrays as Memory Cells

- An array can store primitive elements, such as characters, giving us a compact array.



- An array can also store references to objects.



Compact Arrays

- Defined in “array” module.
- Provides compact storage for arrays of *primitive data types*.
- Example:

```
primes = array('i', [2, 3, 5, 7, 11, 13, 17, 19])
```

Code	C Data Type	Typical Number of Bytes
'b'	signed char	1
'B'	unsigned char	1
'u'	Unicode char	2 or 4
'h'	signed short int	2
'H'	unsigned short int	2
'i'	signed int	2 or 4
'I'	unsigned int	2 or 4
'l'	signed long int	4
'L'	unsigned long int	4
'f'	float	4
'd'	float	8

Array Operations

- `array.insert(i, x)`: add `x` to index `i` $\square O(n)$
- `array.append(x)`: add `x` to end of array.
- `array.remove(x)`: remove first occurrence of `x` $\square O(n)$
- `array.pop(i=-1)`: remove and return `i`th element.
- `array.index(x)`: search and return element `x` in the array.
- `array.tolist()`: return array content in Python List Class.
- ...

Lists

- Built-in list Class
- Denoted with []
- Allow duplicate values
- Allow mixed types
- Index starts with 0

Lists – continued

- Let A = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
- A[0] = "apple"
- A[1] = "banana"
- A[-1]: last item in the list □ "mango"
- A[1:]: new list which starts at index 1
- A[:2]: first two items □ ["apple", "banana "]
- ...

Lists – continued

- `A.append(x)`: Add `x` to the end of the list.
- `A.extend(B)`: Add items in list `B` to the end of list `A`.
- `A.append(B)`: Add list `B` to the end of the list `A`.
- `A.remove(x)`: Remove first occurrence of item `x`.
- `A.pop(i=-1)`: Remove and return `i`th element.
- `del A[i]`: remove `i`th element.

List Comprehension

- Let `A = list(range(10))`
- `B = []`
- for `x` in `A`:
 - `B.append(x**2)`

versus

- `B = [x**2 for x in A]`

List Comprehension – continued

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
```

```
newlist = [x for x in fruits if "a" in x]
```

```
# newlist = ["apple", "banana", "mango"]
```

Tuples

- Built-in list Class
- Cannot be changed
- Denoted with ()
- Allow duplicate values
- Allow mixed types
- Index starts with 0

Tuples – continued

- Let $B = (1, 5, 7, 9, 3)$
- Content cannot be changed
- $B[0] = 1$
- $C = ("A",)$ # tuple with one item

Tuples – continued

- Function returning tuples:

```
def stats(data):  
    return max(data), min(data)
```

```
stats([1, 3, 5, 7, 9]) # returns tuple  
(9, 1)
```

Dictionary

- Built-in list Class
- Denoted with { }
- Allow mixed types
- Key-value pairs
- Unordered
- Dynamic sizing
- Accessed by keys
- Allow dictionary comprehensions

Examples

```
# Iterating through key-value pairs  
for key, value in somedict.items():  
    print(key, ":", value)
```

Sets

- Built-in list Class
- Denoted with { }
- Allow mixed types
- Unordered
- Dynamic sizing
- Allow set comprehensions
- Common set operations, e.g. union, intersection, etc.