


Intor to Python

Lesson 3: Data Structures

List

- A collection of objects
 - Object can have different types
- Items are accessible through their index
 - Index starts from 0
- Items are separated by comma „,“
- Negative index: -1, -2, ...
- Slice: [startIndex : stopIndex]
 - myList[1:3]
 - myList[:3]
 - myList[1:]
- Check membership with „in“/“not in“
 - 1 in myList
 - „Alice“ not in myList



```
1  myList = [1, 2.0, "Ali", True]
2  print(myList[0]) #list index starts from 0 and ends at N-1 (N: number of elements)
3  print(myList[1])
4  print(myList[2])
5  print(myList[3]) #generates an Error!
```

Mutability & Order

- Lists & Strings are both ordered because their elements can be accessed by their position
- `myList[0]`
- `myString = „Ali“`
 - `myString[0] → „A“`
- Lists are mutable
 - `myList[0] = „one“`
- Strings are immutable
 - `myString[0] = „a“ → Error`

List Methods

- `len(myList)` → Number of elements
- `max()`, `min()`
- `sorted()`
- `myList.append()`: add to end
- `myList.insert(pos, element)`
- `+` or `extend()`
- `remove(element)`
 - What happens with duplicate elements?

XGram 1.0

- You are going to develop the next generation of Instagram called „XGram,,.
- For this challenge write 5 lists
 - Users: Alice, Bob
 - postsAlice :Wow! Iam finally in Karlsruhe!, Great music @ AKK
 - postsBob: Great time at python class!
 - imgsAlice: shorturl.at/chDUX, shorturl.at/aorP1
 - imgsBob: shorturl.at/wxFLR
- Add a new message to postBob
 - I just learned about lists!
- add oldPostsAlice to the beginning of postsAlice
 - oldPostsAlice=[„Got the ticket“, „On my way to Karlsruhe“]
- Print last two posts of Alice

Tuples

- Immutable sequence of elements
- Packing & Unpacking data

```
location = (13.4125, 103.866667)
print("Latitude:", location[0])
print("Longitude:", location[1])
```

```
dimensions = 52, 40, 100
length, width, height = dimensions
print("The dimensions are {} x {} x {}".format(length, width, height))
```

Sets

- Mutable unordered collection of data
- Items are unique
- Create a grocery list out of
 - `set(['milk', 'apple', 'orange', 'milk'])`
- `set([1,2,3]) - set([2,3])`
- `intersection()`
- `issubset()`
- `union()`

Dictionaries

- A **dictionary** is a mutable data type that stores mappings of unique keys to values
- `myDict = {key1:value1, key2:value2, ...}`

```
1 myDict = {"Ali":"0176xxxx", "Farshid":"0164xxx"}
2 myDict["Ali"] # "0176xxxx"
3 myDict["Ali"] = "0175xxx"
4 phoneNumber = myDict["Shahin"]
5 phoneNumber = myDict.get("Shahin", "Is not on your contact list")
```


Compound Data Structures

```
elements = {"hydrogen": {"number": 1,
                          "weight": 1.00794,
                          "symbol": "H"},
            "helium": {"number": 2,
                       "weight": 4.002602,
                       "symbol": "He"}}

helium = elements["helium"] # get the helium dictionary
hydrogen_weight = elements["hydrogen"]["weight"] # get hydrogen's weight
```

```
1 doozTable = [['x', 'o', 'o'], ['o', 'x', ' '], [' ', ' ', 'x']]
2 print(doozTable[0])
3 print(doozTable[1])
4 print(doozTable[2])
5 print(doozTable[1][0])
```

	0	1	2
0	X	O	O
1	O	X	
2			X

Dooz 1.0

- Implement dooz for only a single round
- Use ,x' for user1 and ,o' for user2
- User 1 starts the game and enters the row and then enters the column
- User 2 does the same
- Print the dooz table
 - Empty cells with ,, ,,
- Note the type of data returned by input()!
- Note that index starts with zero
- Initial the dooztable with single space strings

XGram 1.1

- Create a compound data structure that stores all data from the 1st version
- Use tuples to pack image urls with the relevant posts
 - Check out the urls to find the relevant posts!
 - For posts that do not have an image set the url to empty string
- Read the name of the user from the input and print the last post including the url (if available)