جَامعة المَنارة
MANARA UNIVERSITY

Introduction to Artificial Intelligence

Lecture 1: Introduction to Python

Lists:

List Specifications:

```
mylist = ["apple", "banana", "cherry", "apple"]
```

- Lists are used to store multiple items in a single variable.
- square brackets []
- List items are ordered, changeable (add/remove elements), and allow duplicate values.
- List items are indexed, the first item has index [0], the second item has index [1] etc.
- List items can be of any data type.

```
list1 = ["apple", "banana", "cherry"]
list2 = [1, 5, 7, 9, 3]
list3 = [True, False, False]
```

- A list can contain different data types.

```
list1 = ["abc", 34, True, 40, "male"]
```

List length:

- len(list_name)

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

Access List Items:

- Through index:

```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

- Range of Indexes:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])
```

Note: The search will start at index 2 (included) and end at index 5 (not included).

Change List Items:

- Change Item Value: To change the value of a specific item, refer to the index number:

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "strawberry"
print(thislist)
```

- Change a Range of Item Values:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "mango"]
thislist[1:3] = ["strawberry", "watermelon"]
print(thislist)
```

Exercise:

1. Declare a list containing the following items: 2, 4, 6, 8, 10
2. Print the length of the list.
3. Print the element at index 3.
4. Change the elements at indices 2, 3 to be 3, 5

Solution:

```
# 1. Declare the list
mylist = [2, 4, 6, 8, 10]

# 2. Print the length of the list
print(len(mylist))

# 3. Print the element at index 3
print(mylist[3])

# 4. Change the elements at indices 2, 3 to be 3, 5
mylist[2:4] = [3, 5]
```

Append Items

- list.append(value)

  Example:

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

Insert Items

- list.insert(index, value)

  Example:

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(2, "watermelon")
print(thislist)
```

Remove List Items

- By value:
  list.remove(value)
  Example:

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

- By index:
  list.pop(index)
  Example:

```
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)
```

Exercise:

1. Declare a list containing the following items: *apple, 5, banana, 7, 9.*
2. Add the following item to the end of the list: *cherry*
3. Add the item *yellow* at index 4.
4. Remove the element at index 2.

Solution:

```
# 1. Declare the list
mylist = ["apple", 5, "banana", 7, 9]

# 2. Add cherry to the end of the list
mylist.append("cherry")

# 3. Add yellow at index 4.
mylist.insert(4,"yellow")

# 4. Remove the element at index 2
mylist.pop(2)

# Print the list to see the changes
print(mylist)
```

Tuples:

Tuple Specification:

```
mytuple = ("apple", "banana", "cherry")
```

- Tuples are used to store multiple items in a single variable.
- Tuples are ordered and **unchangeable** (we can't add, remove, or edit), and allow duplicate values.
- We use round brackets with tuples ().
- Tuple items are indexed, the first item has index [0], the second item has index [1] etc.
- Tuple items can be of any data type.

```
Tuple1 = ("apple", "banana", "cherry")
Tuple2 = (1, 5, 7, 9, 3)
Tuple3 = (True, False, False)
```

- A tuple can contain different data types.

```
Tuple1 = ("abc", 34, True, 40, "male")
```

Tuple Length:

- `len(tuple_name)`

  Example:

```
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))
```

Access tuple elements:

- If I want to access the 2nd element in the tuple:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[1])
```

Exercise:

1. Declare a tuple containing the following items: *'p', 'y', 't', 'h', 'o', 'n'*.
2. Print the length of the tuple
3. Print elements from index 2 to 4 (included).

Solution:

```
# 1. Declare the tuple
mytuple = ('p', 'y', 't', 'h', 'o', 'n')

# 2. Print the length of the tuple
print(len(mytuple))

# 3. Print elements from index 2 to 4 (included)
print(mytuple[2:5])
```

Dictionary:

Dictionary Specifications:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

- Dictionaries are used to store data values in key:value pairs.
- With Dictionaries we use curly brace {}.
- Dictionaries are ordered, changeable.
- Dictionaries don't allow duplicates.

Dictionary Length:

- `len(dictionary_name)`

Example**:**

```
mydict = {
  "name": "Karam",
  "age": "15",
  "ID": 196447
}
print(len(mydict))
```

Access Items:

- `print(mydict["name"]) -> output: "Karam"`

Discovering what keys I have

- `print(mydict.keys())`

Discovering what values I have

- `print(mydict.values())`

Change Items:

- `dictionary_name["key"] = new_value`

Example:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["year"] = 2018
```

Adding Items:

- `dictionary_name["new_key"] = "new_value"`

Example:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["color"] = "red"
```

Removing Item:

- `dictionary_name.pop("key")`

Example:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.pop("model")
print(thisdict)
```

Exercise:

1. Declare a dictionary containing your mobile phone's information, *type, color, RAM*.
2. Print the dictionary.
3. Add an element about the storage of your phone.
4. Remove RAM item.
5. Print the dictionary again.

Solution:

```
# 1. Declare a dictionary containing your mobile phone's information
phone = {
    'type': 'Samsung Galaxy S22',
    'color': 'Black',
    'RAM': 8
}
# 2. Print the dictionary
print(phone)

# 3. Add an element about the storage of your phone
phone['storage'] = 256

# 4. Remove the RAM item
phone.pop('RAM')

# 5. Print the dictionary again
print(phone)
```

Functions:

- def Function_name (arguments):
  content

  Example:

```
def my_function():
  print("Hello from a function")
```

- def Function_name (arguments):
  content
  return result

  Example:
  ```
  def sum_function(a,b):
    return a+b
  ```
- Function call:
  ```
  my_function()
  sum_function(12,8)
  ```

Python input() Function:

The input() function allows user input.
Syntax: input(prompt)
prompt: a string, representing a default message before the input.

Example:
```
x = input('Enter your name:')
print('Hello, ' + x)
```

Exercise:

1. Declare a function that takes three numbers and returns their average.
2. Declare the following variables a, b, and c, and allow the user to enter their values.
3. Apply your function on a,b,c and print the result.
