# Introduction To Python

Dr.Hajialiasgari

Tehran University
Of
Medical Science

November 8, 2024

TEHRAN UNIVERSITY
OF
MEDICAL SCIENCES

1 Python Lists

## What is Not a Collection?

- Most of our variables have one value in them .
- When we put a new value in the variable, the old value is overwritten .

## Code Example

```
$ python
>>> x = 2
>>> x = 4
>>> print(x)
4
```

## A List is a Kind of Collection

- A collection allows us to put many values in a single variable.
- A collection is nice because we can carry all many values around in one convenient package.

## List Example

```
friends = [ 'Joseph', 'Glenn', 'Sally' ]
carryon = [ 'socks', 'shirt', 'perfume' ]
```

## A Character Too Far

- Accessing beyond the end of a string results in an error.
- Be careful when constructing index values and slices.

## List Constants

- List constants are surrounded by square brackets and the elements in the list are separated by commas
- A list element can be any Python object - even another list
- A list can be empty

## Code Example

```
>>> print([1, 24, 76])
[1, 24, 76]
>>> print(['red', 'yellow', 'blue'])
['red', 'yellow', 'blue']
>>> print(['red', 24, 98.6])
['red', 24, 98.6]
>>> print([ 1, [5, 6], 7])
[1, [5, 6], 7]
>>> print([])
[]
```

## Lists and Definite Loop

```
friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends :
    print('Happy New Year:',  friend)
print('Done!')

#output

Happy New Year: Joseph
Happy New Year: Glenn
Happy New Year: Sally
Done!
```

## Looking Inside Lists

- Just like strings, we can get at any single element in a list using an index specified in square brackets

## Code Example

```
>>> friends = [ 'Joseph', 'Glenn', 'Sally' ]
>>> print(friends[1])
Glenn
>>>
```

## Lists are Mutable

- Strings are immutable - we cannot change the contents of a string - we must make a new string to make any change
- Lists are mutable - we can change an element of a list using the index operator

## Code Example

```
>>> fruit = 'Banana'
>>> fruit[0] = 'b'
Traceback
TypeError: 'str' object does not
support item assignment
>>> x = fruit.lower()
>>> print(x)
banana
>>> lotto = [2, 14, 26, 41, 63]
>>> print(lotto)
[2, 14, 26, 41, 63]
>>> lotto[2] = 28
>>> print(lotto)
[2, 14, 28, 41, 63]
```

## How Long is a List?

- The `len()` function takes a list as a parameter and returns the number of elements in the list
- Actually `len()` tells us the number of elements of any set or sequence (such as a string...)

## Len Example

```
>>> greet = 'Hello Bob'
>>> print(len(greet))
9
>>> x = [ 1, 2, 'joe', 99]
>>> print(len(x))
4
>>>
```

## Using the `range` Function

- The `range` function returns a list of numbers that range from zero to one less than the parameter
- We can construct an index loop using for and an integer iterator

## range Example

```
>>> print(range(4))
[0, 1, 2, 3]
>>> friends = ['Joseph', 'Glenn', 'Sally']
>>> print(len(friends))
3
>>> print(list(range(len(friends))))
[0, 1, 2]
>>>
```

## A Tale of Two Loops:

```
friends = ['Joseph', 'Glenn', 'Sally']
for friend in friends :
    print('Happy New Year:',  friend)
for i in range(len(friends)) :
    friend = friends[i]
    print('Happy New Year:',  friend)

#output
Happy New Year: Joseph
Happy New Year: Glenn
Happy New Year: Sally
```

## Concatenating Lists Using +

- We can create a new list by adding two existing lists together

## Concatenating Lists Example

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print(c)
[1, 2, 3, 4, 5, 6]
>>> print(a)
[1, 2, 3]
```

## Lists Can Be Sliced Using :

- Remember: Just like in strings, the second number is up to but not including

## Slicing Example

```
>>> t = [9, 41, 12, 3, 74, 15]
>>> t[1:3]
[41,12]
>>> t[:4]
[9, 41, 12, 3]
>>> t[3:]
[3, 74, 15]
>>> t[:]
[9, 41, 12, 3, 74, 15]
```

## List Methods

```
>>> x = list()
>>> type(x)
<type 'list'>
>>> dir(x)
[... 'append', 'count', 'extend', 'index', 'insert', 'pop', 'remove'
>>>
```

## Building a List from Scratch

- We can create an empty list and then add elements using the append method
- The list stays in order and new elements are added at the end of the list

## Code Example

```
>>> stuff = list()
>>> stuff.append('book')
>>> stuff.append(99)
>>> print(stuff)
['book', 99]
>>> stuff.append('cookie')
>>> print(stuff)
['book', 99, 'cookie']
```

## Is Something in a List?

- Python provides two operators that let you check if an item is in a list
- These are logical operators that return True or False
- They do not modify the list
- `in`
- `not in`

## Code Example

```
>>> some = [1, 9, 21, 10, 16]
>>> 9 in some
True
>>> 15 in some
False
>>> 20 not in some
True
>>>
```

## Lists are in Order

- A list can hold many items and keeps those items in the order until we do something to change the order
- A list can be sorted The `sort` method (unlike in strings) means sort yourself

## Sort Example

```
>>> friends = [ 'Joseph', 'Glenn', 'Sally' ]
>>> friends.sort()
>>> print(friends)
['Glenn', 'Joseph', 'Sally']
>>> print(friends[1])
Joseph
>>>
```

## Built-in Functions and Lists

- There are a number of functions built into Python that take lists as parameters
- Remember the loops we built? These are much simpler.

## Code Example

```
>>> nums = [3, 41, 12, 9, 74, 15]
>>> print(len(nums))
6
>>> print(max(nums))
74
>>> print(min(nums))
3
>>> print(sum(nums))
154
>>> print(sum(nums)/len(nums))
25.6
```

## Best Friends: Strings and Lists

- Split breaks a string into parts and produces a list of strings.
- We think of these as words.
- We can access a particular word or loop through all the words.

## Code Example

```
>>> abc = 'With three words'
>>> stuff = abc.split()
>>> print(stuff)
['With', 'three', 'words']
>>> print(len(stuff))
3
>>> print(stuff[0])
With
>>> print(stuff)
['With', 'three', 'words']
>>> for w in stuff :
...     print(w)
...
With
Three
Words
```

## Splitting by Delimiter

- When you do not specify a delimiter, multiple spaces are treated like one delimiter
- You can specify what delimiter character to use in the splitting

## Code Example

```
>>> line = 'A lot                    of  spaces'
>>> etc = line.split()
>>> print(etc)
['A', 'lot', 'of', 'spaces']
>>>
>>> line = 'first;second;third'
>>> thing = line.split()
>>> print(thing)
['first;second;third']
>>> print(len(thing))
1
>>> thing = line.split(';')
>>> print(thing)
['first', 'second', 'third']
>>> print(len(thing))
3
```

## Remove Specified Index in List

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

#output
['apple', 'cherry']
```

## Remove Specified Item in List

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

#output
['apple', 'cherry']
```

# End of Lists