python™

Week 7

Lists

Programming

- Algorithm
  - A set of rules or steps used to solve a problem
- Data structure
  - A particular way of organizing data in a computer

https://en.wikipedia.org/wiki/Algorithm
https://en.wikipedia.org/wiki/Data_structure

List Constants

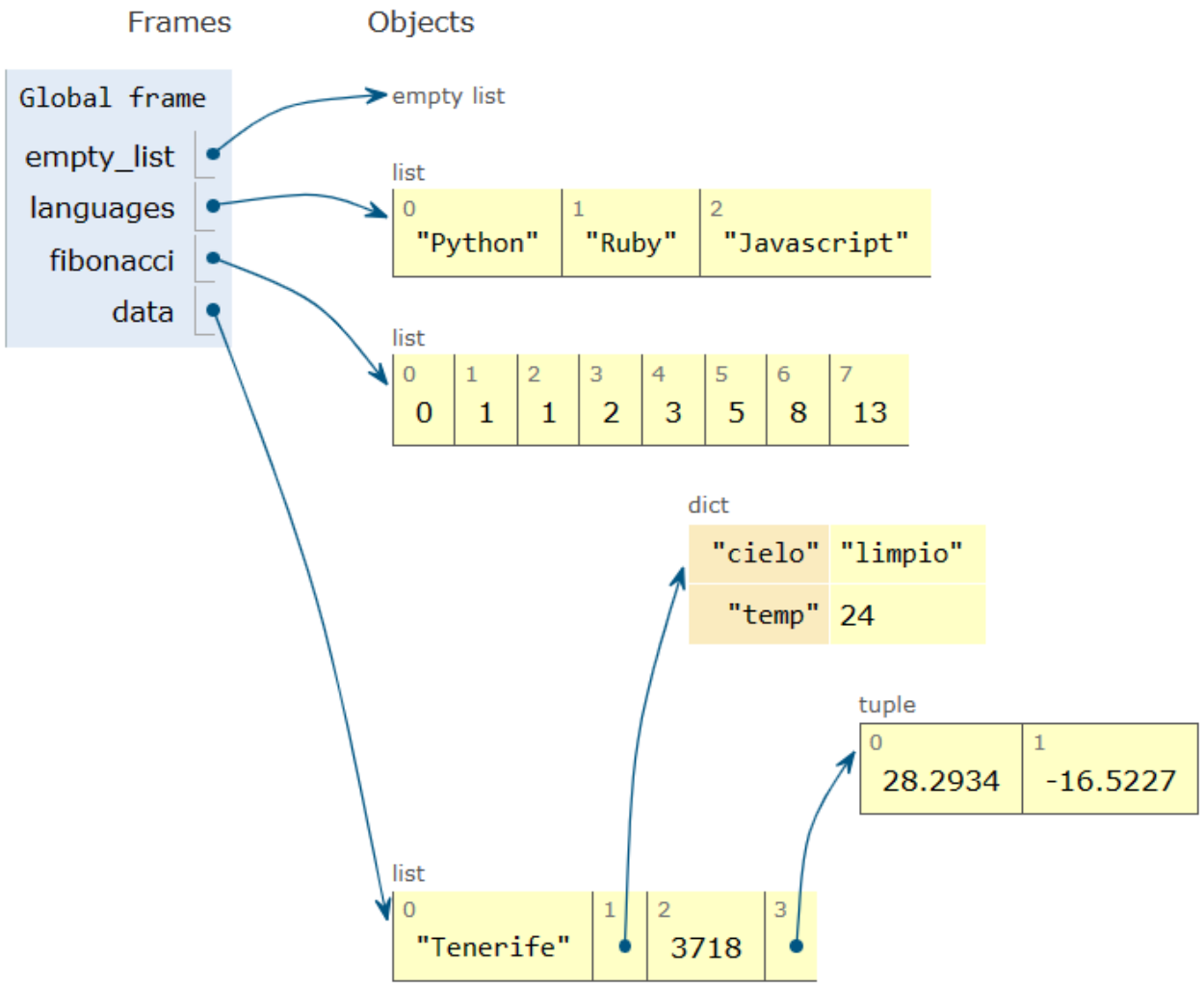
- Lists allow objects to be stored in a defined order and with the possibility of **duplicates**.
- Lists are mutable data structures, which means that we can add, remove, or modify their elements.1 creating lists
- List constants are surrounded by square brackets and the elements in the list are separated by commas
- A list can be **empty**

```
>>> 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([])
[]
```

Creating lists

1. empty_list = []
2. languages = ['Python', 'Ruby', 'Javascript']
3. fibonacci = [0, 1, 1, 2, 3, 5, 8, 13]
4. data = ['Tenerife', {'cielo': 'limpio', 'temp': 24}, 3718, (28.2933947, -16.5226597)]

Understanding List



Conversion to List

- Use the list() function

From a string

```
>>> list('Python')
['P', 'y', 't', 'h', 'o', 'n']
```

From a range

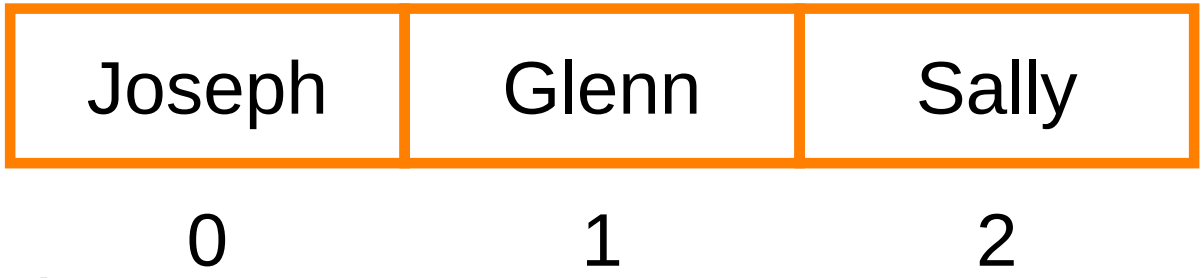
```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Operations with lists

- Access to List
- Add to List
- Delete to List
- Update to List

Access to List

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



- Get an item

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

Slice a list

- List Slicing works completely analogous to string Slicing.
- the second number is "up to but not including"

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

python™

Exercise

- shopping = ['Water', 'Eggs', 'Oil', 'Salt', 'Lemon']

a) shopping[0:3] = ?

b) shopping[10:] = ?

c) shopping[-100:2] = ?

d) shopping[2:100] = ?

Reverse a list

- Option 1:Preserving the original list
  - chopping lists with negative step
  - Using the reversed() function
- Option 2: Modifying the original list
  - Using the reverse() function

```
>>> shopping[::-1]
['Lemon', 'Salt', 'Oil', 'Eggs', 'Water']

>>> list(reversed(shopping))
['Lemon', 'Salt', 'Oil', 'Eggs', 'Water']
```

```
>>> shopping.reverse()
['Lemon', 'Salt', 'Oil', 'Eggs', 'Water']
```

Exercise

- shopping = ['Water', 'Eggs', 'Oil', 'Salt', 'Lemon']

a) shopping[0:3] = ?

b) shopping[10:] = ?

c) shopping[-100:2] = ?

d) shopping[2:100] = ?

Add to end of list

- One of the most used operations in lists is to add elements to the end of them. Python offers us the append() function

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

python™

Example

- We want to build a list with the even numbers of [0, 20)?

```
>>> even_numbers = []

>>> for i in range(20):
...         if i % 2 == 0:
...             even_numbers.append(i)
...
>>> even_numbers
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

python™

Built-in Functions and Lists

- There are a number of functions built into Python that take lists as parameters:
  - len, max, min, sum, sort

```
>>> 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
```

List Python Program

```
total = 0
count = 0
while True :
    inp = input('Enter a number: ')
    if inp == 'done' : break
    value = float(inp)
    total = total + value
    count = count + 1

average = total / count
print('Average:', average)
```

```
Enter a number: 3
Enter a number: 9
Enter a number: 5
Enter a number: done
Average: 5.66666666667
```

```
numlist = list()
while True :
    inp = input('Enter a number: ')
    if inp == 'done' : break
    value = float(inp)
    numlist.append(value)

average = sum(numlist) / len(numlist)
print('Average:', average)
```

Modify a list

- Using index

```
>>> shopping = ['Water', 'Eggs',
'Oil']
>>> shopping[0]
'Agua'
>>> shopping[0] = 'Jugo'
>>> shopping
 ['Juice', 'Eggs', 'Oil']
```

```
>>> shopping[100] = 'Chocolate'
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
IndexError: list assignment index
out of range
```

Modify a list

- Using slice, assign values to chunks of a list

```
>>> shopping = ['Water', 'Eggs', 'Oil', 'Salt', 'Lemon']
>>> shopping[1:4]
['Eggs', 'Oil', 'Salt']
>>> shopping[1:4] = ['Tuna', 'Pasta']
>>> shopping
['Water', 'Tuna', 'Pasta', 'Lemon']
```

python™

Delete items

- By its index: Through the **del** command:

```
>>> shopping = ['Water', 'Eggs', 'Oil', 'Salt', 'Lemon']
>>> del shopping[3]
>>> shopping
['Water', 'Eggs', 'Oil', 'Lemon']
```

Delete items

- By its value: Using the remove() function
- If there are duplicate values, the remove() function will only delete the first occurrence.

```
>>> shopping = ['Water', 'Eggs', 'Oil', 'Salt', 'Lemon']
>>> shopping.remove('Salt')
>>> shopping
['Water', 'Eggs', 'Oil', 'Lemon']
```

Delete items

- By its index (with extraction)
- he pop() function that, in addition to deleting, "retrieves" the element

```
>>> shopping = ['Water', 'Eggs', 'Oil', 'Salt', 'Lemon']
>>> product = shopping.pop() # shopping.pop(-1)
>>> product
Lemon
>>> shopping
['Water', 'Eggs', 'Oil', 'Salt`]
>>> product = shopping.pop(2)
>>> product
'Oil
>>> shopping
>>> shopping
['Water', 'Eggs', ' 'Salt ']
```

Delete items

- By rank: By chopping lists

```
>>> shopping = ['Water', 'Eggs', 'Oil', 'Salt', 'Lemon']
>>> shopping[1:4] = []
>>> shopping
????
```

String and Lists

- Convert list to text string


```
>>> abc = 'With three words'
>>> stuff = abc.split()
>>> print(stuff)
['With', 'three', 'words']
>>> print(len(stuff))
3
>>> print(stuff[0])
With
```

```
>>> shopping = ['Water', 'Eggs', 'Oil',
'Salt', 'Lemon']
>>> '|'.joint(shopping)
>>> shopping
'Water', 'Eggs', 'Oil', 'Salt', 'Lemon'
```

python™

Sort a list

- Keeping original list

```
>>> shopping = ['Water', 'Eggs', 'Oil', 'Salt', 'Lemon']
>>> sorted(shoping)
['Eggs', 'Lemon', 'Salt', 'Water']
```

- Modifying the original list

```
>>> shopping = ['Water', 'Eggs', 'Oil', 'Salt', 'Lemon']
>>> shooping.sorted()
>>> shopping
['Eggs', 'Lemon', 'Salt', 'Water']
```

python™

Iterate over a list

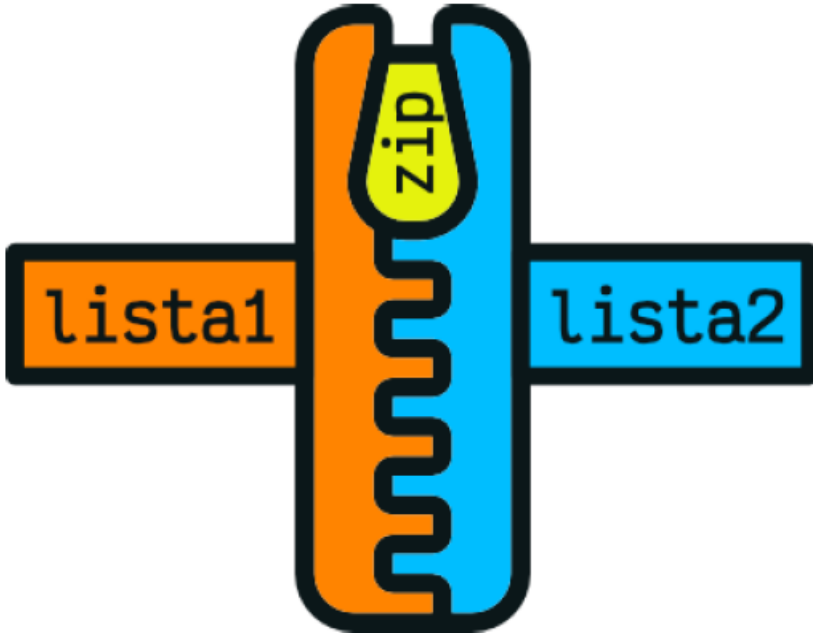
- Keeping original list

```
>>> shopping = ['Water', 'Eggs', 'Oil', 'Salt', 'Lemon']
>>> for product in shopping:
...         print(product)
 Water
 Eggs
 Oil
 Salt
 Lemon
```

- Modifying the original list

```
>>> shopping = ['Water', 'Eggs', 'Oil', 'Salt', 'Lemon']
>>> for i, product in enumerate(shopping):
...         print(i, product)
0 Water
1 Eggs
2 Oil
3 Salt
4 Lemon
```

Iterate over multiple lists



python™

Iterate over multiple lists

```
>>> shopping = ['Water', 'Oil', 'Rice']
>>> details = ['natural mineral', 'virgin olive', 'basmati'])
>>> for product, detail in zip(shopping, details):
...         print(product, detail)
Water natural mineral
Oil virgin olive
Rice basmati
```

```
>>> shopping = ['Water', 'Oil', 'Rice']
>>> details = ['natural mineral', 'virgin olive', 'basmati'])
>>> list(zip(shopping, details))
[('Water', 'natural mineral'), ('Oil', 'virgin olive'), ('Basmati rice')]
```

python™

Exercise

- Known a numerical string, to create the list with those values that start with the digit 4:
- Input: '32,45,11,87,20,48‘
- Output: [45, 48]

Exercise

- Create a list containing the result of applying the function $f(x) = 3x + 2$ for $x \in [0, 20)$.

Exercise

- Write a program that allows you to multiply only matrices with 2 rows by 2 columns.
- Let's see a concrete example:

  A = [[6, 4], [8, 9]], B = [[3, 2], [1, 7]]