

PYTHON OPEN LAB

LIST,
DICTIONARY,
STRING

GitHub

- Why github?
 - No access restriction with public repository.
- Easy to clone repository
 - `git clone xxxx`
- Easy to update files
 - `git pull`

Why we need to learn these strange types?

- Can we finish all jobs with just int, float, bool, string?
 - Maybe, but it will cost you a lot of time when you meet something very complicated
 - ex: get the average age of all Columbia students
 - ageStu1, ageStu2, ageStu3, ageStu4, ageStu5.... ageStui
- We need powerful tools
 - List, dictionary, string
 - List, dictionary, string are object types

What is object type?

- We will talk about it in later parts
- For now, we can see it as a group
 - It contains child elements
 - We can operate on its child elements

List

- `List = [1, 2, 3]` — declare a list
- This is what we learned from last week's session.
- What if I want to add elements to the end this list?
 - `List.append(4);`
 - `print(List)`
 - console: `[1, 2, 3, 4]`

List

- `List = [1, 2, 3]`
- What if I want to add element between 1 and 2?
 - `List.insert(1,0)`
 - `print(List)`
 - console: `[1, 0, 2, 3]`

List

- `List = [1, 2, 3]`
- What if I want to get the element in the position 1?
 - `List[1]`
 - console: 2
- What if I want to change element in the position 1 to 100?
 - `List[1] = 100`
 - `print(List[1])`

List

CONTINUED

- `List = [1, 2, 3]`
- What if I want to get the element in the position 1?
 - `List[-2]`
 - console: 2
 - `List[-1]`
 - console: 3
- Negative index: read from right side

List

- `List = [1, 2, 3]`
- Get the length of this list
 - `len(List)`
 - console: 3
 - `List.append(0)`
 - `len(List)`
 - console: 4

List

- Get the elements between position i and position j
 - `List[i , j+1]`
 - j+1 is exclusively included!
 -

List

- `List = [1,2,3,4,5,6]`
- Get elements from position 2 to the end
 - `List[2 : 6]`
 - A smarter way:
 - `List[2 : len(List)]`
 - A more advanced way:
 - `List[2 :]`

List

- `List = [1,2,3,4,5,6]`
- Get elements from the start to position 4
 - `List[0 : 5]`
 - A more advanced way:
 - `List[: 5]`

List

- Operation List
 - `append(x)` — add element to the end of the list
 - `insert(i, x)` — add element to the specific position of list
 - `count(x)` — get the times of x appear in the list
 - `remove(x)` — remove the first x in the list
 - `sort()` — sort the list
 - `extend(List b)` — extend b to the end of present list

List

- An amazing part of list is that the elements of List can also be lists
- One - dimension list
 - List = [1,2,3]
- Two - dimension list
 - List = [[1,2,3], [4,5,6], [7,8,9]]

List

- `List = [[1,2,3], [4,5,6], [7,8,9]]`
- See it this way:
 - First row: [1,2,3]; second row: [4,5,6]; third row: [7,8,9]
 - First column: [1,4,7]; second column:[2,5,8]; ...
- See it this way:
 - To get the element in the *i* row and *j* column
 - `row[i-1][j-1]`

List

- `List = [[1,2,3], [4,5,6], [7,8,9]]`
- Get 2: `List[0][1]`
- Get 5: `List[1][1]`
- Get 8: `List[2][1]`
- To prove that each element in this 2D list is a list
 - `List[0] — [1,2,3]`
 - `List[1] — [4,5,6]`
 - `type(List[0])`

List

- What have we learned so far?
 - Manipulate the elements in list
 - Still many functions unlearned
 - Learning Python(5th edition)
- Time for a practice
 - Manipulate a string list
 - Put "I" , "love", "Columbia" to a list

Dictionary

- Most flexible built-in data types in Python
- Difference between list and dictionary
 - list: ordered collections of objects
 - dictionary: unordered collections
- How to fetch elements in dictionary?
 - Items are stored and fetched by key, instead of by positional offset

A scenario to apply dictionary

- In an exam
 - Mike gets a score of 80
 - Emily gets a score of 82
 - Kevin gets a score of 85
 - Jeff gets a score of 90
- I want to store them and get students score according to name
 - It can be done by lists

Dictionary

- `dictionary = {}` - declare a dictionary
 - `dictionary["Mike"] = 80`
 - `dictionary["Emily"] = 82`
 - `dictionary["Kevin"] = 85`
 - `dictionary["Jeff"] = 90`
- We have finished storing elements to a dictionary
- Let's see the form of a dictionary: print it!

Dictionary

- Now with this dictionary, I want to know the score of students
- Fetch values by key
 - `dictionary["Mike"]`
 - 80
 - `dictionary["Jeff"]`
 - 90

Dictionary

- The most important thing about dictionary is that keys can not be the same for any two pairs
- Two "Mike"?
 - `dictionary["Mike"] = 80`
 - `dictionary["Mike"] = 90`
 - `print(dictionary["Mike"])`

Dictionary

- The keys of dictionary is a set
- What is a set?
 - A group only contains unique elements.
- What is the principle of dictionary?
 - Hash-function
 - It can be implemented by a 2D - list

Dictionary

- Key: value pair
 - Value can be int, float, bool, string
 - Value can also be a list, a dictionary — the dictionary is very flexible!
 - `dictionary["class1"] = ["Mike","Jeff","Emily"]`
 - `dictionary["scores"] = [90,80,85]`

Dictionary

CONTINUED

- `dictionary = {}`
- `dictionary["ScoreOfClass1"] = {}`
- `dictionary["ScoreOfClass1"]["Mike"] = 85`
- `dictionary["ScoreOfClass1"]["Emily"] = 90`
- `dictionary["ScoreOfClass1"]["Jeff"] = 95`
- `print(dictionary["ScoreOfClass1"])`

Dictionary

- `dict = {}`
- `dict[99] = 100`
- `dict["99"] = 100`
- Not suggested

String

- Double quotes and single quote
 - 'hello world'
 - "hello world"
- Anything between double quotes and a single quote is a string
 - Not only alphabet is string
 - " *&(&(*)*)" "
 - '1111111111'

String

- Question
 - `A = '''python open lab'''`
 - What is A?
- Attention
 - `A = '''python open lab'`
 - Brackets must match!
 - `A = '''python open lab''`

String

- Concatenation

- $S1 = \text{"hello "}$
- $S2 = \text{"world"}$
- $S3 = S1 + S2$
- What is S3?
- What is S1, S2?

- Index

- $A = \text{"12345"}$
- $A[0]$
- $A[4]$
- $A[-2]$
- $A[-1]$

String

- Slice — really similar to that of list
- `A = "12345678"`
- `A[2:5]` from index 2 to index 4(index 5 is not included!)
- `A[5:-1]` Will this work?
- `A[2 :]`
- `A[: 7]`

String

- Try examples on your Jupyter notebook!
- Next week we will talk more about string.
- And we will learn about loop in list, dictionary and string!