Chapter 9.
Dictionary

Starting out with Python

Code Examples with Jupyter Lab

Kyu Lee. Ph. D.

Computer Science

Dictionary

Goal of this chapter

```
dict1 = { 1:'A',    2:'B',  3:'C'}
```

- How to construct dictionary
- How to get a value with a particular key
    - Add a value with key
- How to delete an item from the dictionary
- How to traverse all items in a dictionary
- What happen when we use the wrong (non-existing) key

Dictionary

- dictionary
  - key - value pair

```
dict1 = { 1:'A',    2:'B',  3:'C'}
```

Dictionary: Key → Value

1	'A'
2	'B'
3	'C;

Key : immutable object

Value : any time

Retrieving a value from a dictionary

- dictionary

dict1 = { 1:'A', 2:'B', 3:'C'}

Dictionary: Key → Value

1	'A'
2	'B'
3	'C;

```
print (dict1[1])

print (dict1[0])      # Error? Key Error.
                      # use the try-except pattern
```

Retrieving a value from a dictionary

- Key Check
  - in
  - not in

dict1 = { 1:'A', 2:'B', 3:'C'}

```
if  4 not in dict1 :
      print ('4 is not a key')
```

Adding a value to a dictionary

```
dict1 = { 1:’A’,    2:’B’,  3:’C’}

dict1[4] = ‘D’
```

Iterate over a dictionary

- Methods from Dictionary
  - keys()
  - items()
  - values()

```
scores_record = {        'Kim':[100,80,70,60],
                        'Bill':[100,90,80,60],
                        'Mary':[90,80,70,100]}

for key in scores_record.keys():
        print (scores_record[key])
```

Iterate with the keys

```
for value in scores_record.values():
        print (value)
```

Iterate with the values

```
for key, value in scores_record.items():
        print (key)
        print (value)
```

Iterate with the key, values

Dictionary Methods

clear()	Clears the contents of a dictionary.
get(key, default)	dictionary.get(key, default) Gets the value associated with a specified key. If the key is not found, the method does not raise an exception. Instead, it returns a default value.
items()	Returns all the keys in a dictionary and their associated values as a sequence of tuples.
keys()	Returns all the keys in a dictionary as a sequence of tuples.
pop(key, default)	pop(key, default) Returns the value associated with a specified key and removes that key-value pair from the dictionary. If the key is not found, the method returns a default value.
popitem()	Returns a randomly selected key-value pair as a tuple from the dictionary and removes that key-value pair from the dictionary.
values()	Returns all the values in the dictionary as a sequence of tuples.

Dictionary Methods

- Methods from Dictionary
  - **fromkeys( )**
    - `dict.fromkeys(keys, value)`
    - returns a dictionary with the specified keys and the specified value
    - **All keys have a same value**

```
keys = [1,2,3]
values = ['Kim', 94598, '1 Main Street']

dict1 = dict.fromkeys(keys,values)

print (dict1)
```

{1: ['Kim', 94598, '1 Main Street'], 2: ['Kim', 94598, '1 Main Street'], 3: ['Kim', 94598, '1 Main Street']}

Deleting a value from a dictionary

```
dict1 = { 1:'A',    2:'B',  3:'C'}
```

```
del dict1[1]
```

When you know the key

Deleting a value from a dictionary

Deleting by value

delete an item that has value ‘B’

Find an item that has value ‘B’ and then delete it

option 1)

```
#delete an item that has the value 'B'
for k, v in dict(dict1).items():
        if v == 'B':
                del dict1[k]
print (dict1)
```

option 2)

```
dict1 = { key:value for key, value in dict1.items() if dict1[key] != 'B'}
```

Length of elements

```
dict1 = { 1:’A’,    2:’B’,  3:’C’}

len(dict1)    # 3
```

Lab 1: Dictionary Example 1

- Create a dictionary **student** to manage the student's scores
  - key : student's name
  - value: **list of scores**
- [Task 1] Make a function **makeStudent(names, scores)**
  - Construct the dictionary **student** with the given list values
  - **return** the dictionary **student**
- [Task 2] Make a function **printStudent(student)**
  - Print all dictionary information with key and values
  - no return value
- [Task 3] Make a function **getMaxScore(student)**
  - find the student who has the greatest summation of scores
  - return the student's name

```
Given list value
names = [ Kim, Bill, Mary]
scores =[ [100,80,70,60],
          [100, 90, 80, 60],
          [90, 80,70, 100] ]
```

Key	value			
Kim	100	80	70	60
Bill	100	90	80	60
Mary	90	80	70	100

Mixing Data Type in a Dictionary

```
mdict1 = { 1:100, 'Scores':[10, 20, 30], 'Name':'Junior', 'Course':['CS1','CS2','CS3']}

print (mdict1)
```

Dictionary Lab 2

- Copy a dictionary from the existing dictionary with subset of keys
  - Copy a dictionary with key values.

```
emp_dict = {
    "name": "Kelly",
    "age": 25,
    "salary": 8000,
    "city": "New york"}
```

- Make a function copyDict(d, k)
  - d: dictionary
  - k: list of keys to be copied
  - **return** the copied dictionary

```
keys = ["name", "salary"]
```

```
copy_dict = {
    "name": "Kelly",
    "salary": 8000 }
```

Dictionary Lab 2

- Create a dictionary from the existing dictionary with subset of keys

```
emp_dict = {
    "name": "Kelly",
    "age": 25,
    "salary": 8000,
    "city": "New york"}
```

```
keys = ["name", "salary"]
```

```
new_dict = {
    "name": "Kelly",
    "salary": 8000 }
```

```
new_dict = {}
for k in keys:
        try:
                new_dict[k] = emp_dict[k]
        except KeyError:
                pass
print (new_dict)
```

```
new_dict = { k:emp_dict[k] for k in keys}        # simple , but there is no error check
```

Dictionary Lab 3

- Make a function deleteDictItems(d, k)
  - d: original dictionary
  - k: list of keys to be deleted
  - return the deleted dictionary

```
emp_dict = {
    "name": "Kelly",
    "age": 25,
    "salary": 8000,
    "city": "New york"}
```

```
keys = ["age", "city"]
```

```
emp_dict = {
    "name": "Kelly",
    "salary": 8000 }
```

Dictionary Lab 3

POP

- Delete a list of keys from a dictionary

```
emp_dict = {
    "name": "Kelly",
    "age": 25,
    "salary": 8000,
    "city": "New york"}
```

```
keys = ["age", "city"]
```

```
emp_dict = {
    "name": "Kelly",
    "salary": 8000 }
```

```
for k in keys:
    emp_dict.pop(k)      # pop() returns a value and delete the item at the same time
```

What if the k does not exist
You should check the key value and then try to pop()

```
emp_dict = { k:emp_dict[k] for k in emp_dict.keys() - keys}
```

Dictionary Lab 4

- Change the ‘org_keyval’ to the ‘new_keyval’
- Make a function changeKey(org_dict, org_keyval, new_keyval)
  - org_dict
  - org_keyval
  - new_keyval

‘city’

‘location’

```
emp_dict = {
    "name": "Kelly",
    "age": 25,
    "salary": 8000,
    "city": "New york"}
```

In this function,
Change the key value from “city” to “location”

```
emp_dict = {
    "name": "Kelly",
    "age": 25,
    "salary": 8000,
    "location": "New york"}
```

Dictionary Lab 4

- Rename a key
  - city changes to → location

```
emp_dict = {
    "name": "Kelly",
    "age": 25,
    "salary": 8000,
    "city": "New york"}
```

```
emp_dict['location'] = emp_dict.pop('city')
```

Asterisk in python

*

**

Asterisk

- * single asterisk : unpacking

```
lst = [10, 20, 30]

print (lst)      # [10, 20, 30]
print (*lst)     # 10, 20, 30
```

```
l1 = [1,2,3]
l2 = [4,5,6]

l1.append(l2)
print (l1)
```

```
l1 = [1,2,3]
l2 = [4,5,6]

l3 = [l2]
print (l3)
l3 = [*l2]
print (l3)
```

```
l1 = [1,2,3]
l2 = [4,5,6]
l3 = [*l1, *l2]
print (l3)
```

test each code box and see the differences

**

- Positional keyword argument

```
def fn(**kwargs):
        for k, v in kwargs.items():
                print (k, v)

dict1 = {'a':1, 'b':2, 'c':3}
# dict1 = {1:'a', 2:'b', 3:'c'}

fn(**dict1)
fn(name='Kim', score=100, address='94598')
```

using **

- Merge two dictionary
  - Try to test two code segments

```
dict1 = {'name': 'KIM', 'ZIP':94598,
'address':'1234 Grand ave'}
dict2 = {'score':[100,90], 'Grade':'Senior'}
dict3 = {**dict1, **dict2}

print (dict3)
```

```
dict3 = dict1.copy()
dict3.update(dict2)
print (dict3)
```

Lab 5: using **

```
dict1 = {'name': 'KIM', 'ZIP':94598, 'address':'1234 Grand ave'}
```

- Merge two dictionary
- Make the function **mergeDict(dict1, dict2)**
  - dict1: dictionary 1
  - dict2: dictionary 2
  - return the merged new dictionary dict3

```
dict2 = {'score':[100,90], 'Grade':'Senior'}
```

```
dict3 = {'name': 'KIM', 'ZIP':94598, 'address':'1234 Grand ave',
'score':[100,90], 'Grade':'Senior'}
```

Dictionary
Lab 6: Review of the dictionaries

Lab 6

	A	B	C	D	E
1	name	ID	Math	English	Computer
2	John Doe	2022-0002	100	90	90
3	Jane Doe	2023-0001	70	100	90
4	Mary Smith	2023-0002	100	100	100
5	Bill Watson	2023-0003	100	75	85
6					

students.xlsx

- The purpose of Lab
- Step 1: Understanding the list of dictionary
  - Make a **list of dictionary**
- Step 2: Learn how to read lines from excel file
  - Learn DataFrame from ‘pandas’ to read the excel file
- Step 3: Learn how to construct the dictionary and append it to the list
  - The thinking way how to construct a list of dictionary from the excel file
- Step 4:
  - From the list of dictionary, find a particular value by the key

Lab 6 :

- Various ways to read Excel files
  - “openxypl”, “xlrd”, and “Pandas”
- We will use “pandas” for this lab
  - This slide shows the other ways to read Excel files

modules for reading the Excel file

- csv
- xlrd, example code
- **openpyxl**
- pandas

download it to use in your code.
Place it in the same directory.

```
import xlrd

wb = xlrd.open_workbook('students.xlsx')
ws = wb.sheet_by_index(0)
rows = []

print (ws.nrows)

keys = ws.row_values(0)
print (keys)


for rownumber in range(1, ws.nrows):
        print (ws.row_values(rownumber))
#       row = ws.row_values(rownumber)
#       row[0], row[1], row[2]
```

```
import openpyxl


filename = 'students.xlsx'
wb = openpyxl.load_workbook(filename)


ws = wb.active


for row in ws.iter_rows():
    for cell in row:
        print (cell.value, end = '\t\t')
    print ()
```

Lab 6

- 10 mins to pandas
  - https://pandas.pydata.org/docs/user_guide/10min.html
- Dataframe from Pandas
  - Example Source Code
  - Click the link to see the source code and Run all code cells in this Jupyter notebook file.
    - Figure out how to access the row and column values in DataFrame

```
import pandas as pd


df = pd.read_excel('students.xlsx')
df
```

	name	ID	Math	English	Computer
0	John Doe	2022-0002	100	90	90
1	Jane Doe	2023-0001	70	100	90
2	Mary Smith	2023-0002	100	100	100
3	Bill Watson	2023-0003	100	75	85

```
df.columns      # heading
df.shape        # # of rows and columns
df.shape[0]     # # of rows
df.shape[1]     # # of columns

df.iloc[0]      # the first row
df['ID']        # column 'ID' values
```

Lab 6

- (1) Create a function `makeStudentDictionary( )`
  - Parameter:
    - None
  - Task
    - Open a "Students.xlsx" file
    - Read a Excel file and make a "DataFrame"
      - Construct a dictionary from a "DataFrame".
      - See the Example Source.
  - return value
    - List of dictionaries for all students information

```
[{'name': 'John Doe', 'ID': '2022-0002', 'Math': 100, 'English': 90, 'Computer': 90},
{'name': 'Jane Doe', 'ID': '2023-0001', 'Math': 70, 'English': 100, 'Computer': 90},
{'name': 'Mary Smith', 'ID': '2023-0002', 'Math': 100, 'English': 100, 'Computer': 100},
{'name': 'Bill Watson', 'ID': '2023-0003', 'Math': 100, 'English': 75, 'Computer': 85}]
```

```
name -> John Doe        ID -> 2022-0002        Math -> 100        English -> 90
name -> Jane Doe        ID -> 2023-0001        Math -> 70         English -> 100
name -> Mary Smith      ID -> 2023-0002        Math -> 100        English -> 100
name -> Bill Watson     ID -> 2023-0003        Math -> 100        English -> 75
```

Lab 6

- (2) Create a function `printStudents(list of dictionary)`
  - Parameter
    - list of dictionary that was returned from the function `makeStudentDictionary( )`
  - Print all dictionary values with the user-friendly output format
  - return value
    - none

Lab 6

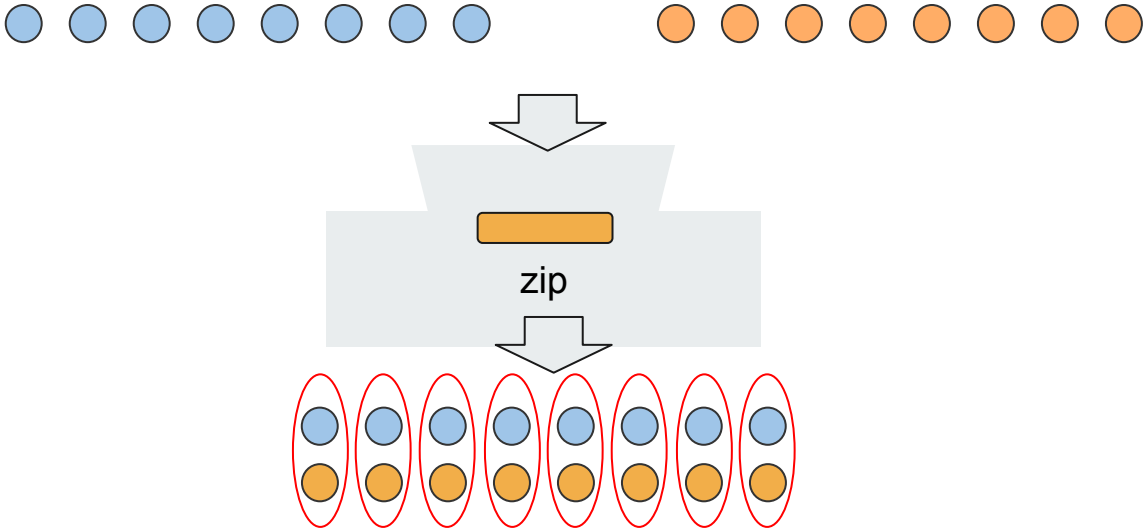
- (3) Create a function `findStudent(list of dict, int)`
  - Parameters:
    - list of dictionaries
    - one integer value for ID
  - Find the students who has the given ID in the parameter
  - Return value
    - a score list of the student who has the given ID

```
[70, 100, 90]
```

zip

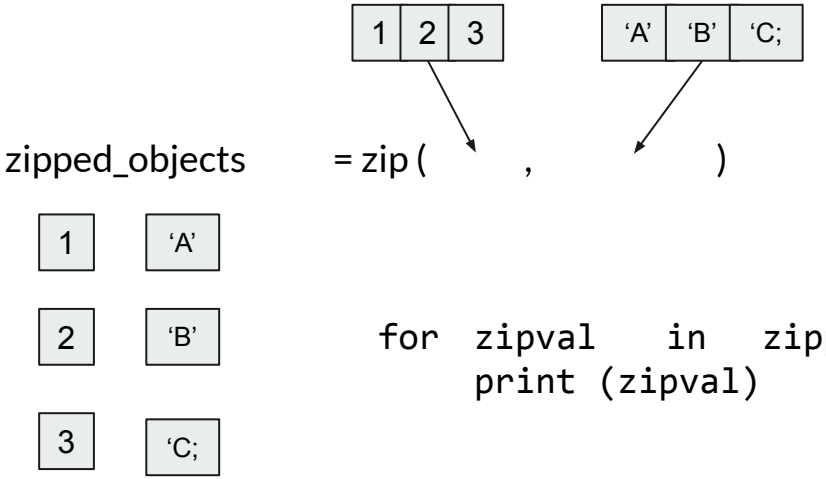
zip

- zip
  - takes Iterable objects or containers
  - return a single iterator, having mapped values from all the containers



zip

- zip
  - takes Iterable objects or containers
  - return a single iterator, having mapped values from all the containers




```
for zipval in zipped_objects:
    print (zipval)
```

zip

Lab 7

- The list of student's names = ['Bill', 'John', 'Kurt']
- The list of score values = [100, 90, 90]

Make a zip statement and print the paired values with a zip object

zip

Lab 8

- The list of course ID =[1001, 1002, 1003]
- The list of course Name = ['C Programming', 'Java Programming', 'Python Programming']

Make a zip statement and print the paired values with a zip object

https://github.com/LPC-CSDept/CS7L98

Lab 9 : zip to dictionary

- Write a function `makeDict(heading, valueset)` to construct a dictionary from two list values 'heading' and 'valueset'
- **Return** value:
  - a list of **dictionary** made with two list values
- **Requirement**
  - Use `zip( )`

Make a list of dictionaries from two lists

```
heading = ['id', 'name', 'address']

valueset = [ [10, 'Kim', '123 Main'],
             [20, 'Bill', '345 Grand'],
             [30, 'Mary', '123 Blvd']
           ]
```

Tip: Try this statement
`dict(zip(heading, valueset[0]))`

heading

ID	Name	Address

valueset

10	Kim	123 Main
20	Bill	345 Grand
30	Mary	123 Blvd

Return value

```
[{'id': 10, 'name': 'Kim', 'Address': '123 Main'},
 {'id': 20, 'name': 'Bill', 'Address': '345 Grand'},
 {'id': 30, 'name': 'Mary', 'Address': '123 Blvd'}]
```

* operator in zip ( )

to unzip

Lab 10: zip(*)

```
# Student's information: ID, Name, GradeLevel, Zip
student_list = [ [1001, 'Bill', 'Senior', 94568],
                 [1002, 'Kurt', 'Junior', 94598],
                 [1003, 'Kim', 'Senior', 94598] ]

print (student_list)

for value in zip(*student_list):
        print (value)

IDlist, Namelist, Glist, Zlist = zip(*student_list)

print (IDlist)
print (Zlist)
```

zip(, ,)

zip( )

vs

```
for v in zip(student_list):
        print (v)
```

```
([1001, 'Bill', 'Senior', 94568],)
([1002, 'Kurt', 'Junior', 94598],)
([1003, 'Kim', 'Senior', 94598],)
```

zip(*)

- Can we unzip a dictionary?
- Can we unzip a non-zip object?
  - such as list, dictionary, and tuple

Lab 10 : zip(*)

- Unzip Dictionary ?
  - use dictionary.items()

```
dictionary1 = { 10:"Kurt", 20:"Jim", 30:"Bill"}
print (dictionary1)

unzipped = zip(*dictionary1.items())

for values in unzipped:
        print (values)

IDlist, Namelist = zip(*dictionary1.items())
print (IDlist)   # tuple
print (Namelist) # tuple

IDlist?
```

```
dictionary1 = { 10:"Kurt", 20:"Jim", 30:"Bill"}
print (dictionary1)

IDlist, Namelist = zip(*dictionary1.items())
print (IDlist)   # tuple
print (Namelist) # tuple

IDlist?

for v in zip(*Namelist):
        print (v)
('K', 'J', 'B')
('u', 'i', 'i')
('r', 'm', 'l')
```

Lab 11: zip(*)

- Write a function `getColumn(numbers)` to take a list of list numbers as a parameter and returns a new list of lists representing the column values from the original input.
  - For example, [[10, 40, 70, 100], [20, 50, 80, 110], [30, 60, 90, 120]]
- **Requirement**
  - use **zip(*), see page 42 and 43**

```
numbers = [ [10, 20, 30],
            [40, 50, 60],
            [70, 80, 90],
            [100, 110, 120] ]
```

```
returnval = [[10, 40, 70, 100],
             [20, 50, 80, 110],
             [30, 60, 90, 120]]
```

zip(*)

- Unzip list ?

```
namelist = ['Kim', 'Bill', 'Kurt']


for v in zip(*namelist):
        print (v)
('K', 'B', 'K')
('i', 'i', 'u')
('m', 'l', 'r')

# What about integer list?
IDlist = [10, 20, 30]
zip(*IDlist) # Error. why?
```

zip(*)

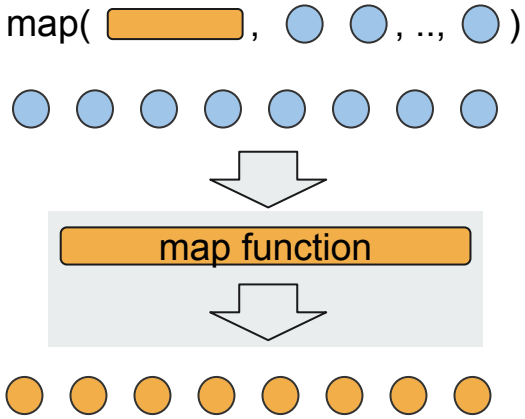
- unzip list of list ?

```
list1 = [ [10, 20, 30], ['Kim', 'Jim', 'Sam']]

for value in zip(*list1):
        print (value)
```

map

map

- map() is used to
  - apply a **function** on **all elements** of a specified iterable
  - return a **map object**
- map(func, [...])




```
def    square(val):
       return val * val

lst = [1,2,3,4,5]

sqrmap = map(square, lst)

for v in sqrmap:
       print (v)
```

map and lambda

- lambda function can be used in map()

```
sqr = lambda val : val * val

lst = [1,2,3,4,5]

sqrmap = map(sqr, lst)

for v in sqrmap:
       print (v, end=' ')
```

or

```
lst = [1,2,3,4,5]

sqrmap = map(lambda val : val * val, lst)

for v in sqrmap:
       print (v, end=' ')
```

map and list

- list as a map function

```
lst1 = list(‘python’)      # lst1 = [‘p’, ‘y’, ‘t’, ‘h’, ‘o’, ‘n’ ]
```

```
list('sat')

mylst = ['C++', 'Python', 'Java']

mapobj = map(list, mylst)
```

Lab 12 : map

- Write a function halfValue(numbers) that uses the map() function to divide each element in a list by 2.
  - Truncate the value when there is a fractional value
    - 5 // 2 = 2
  - Save your result as a list and return it

```
numbers = [ 10, 20, 30, 40, 50]


# Expected return value
[5, 10, 15, 20, 25]
```

Lab 13 : map

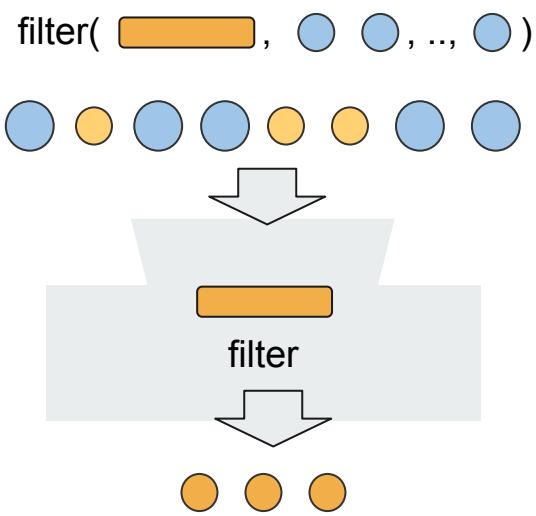
- `Write a function setOddNumber(numbers) that uses the map() function to set to 1 if the element value is an odd number, otherwise set to 0`
  - `Save your result as a list and return it`
- Requirement: use `map`
  - to modify a list, setting all even elements to 0 and all odd elements to 1.

```
mylist = [ 5, 10, 15, 20, 21, 25, 27]


# Expected output
[1, 0, 1, 0, 1, 1, 1]
```

filter

filter

- Filter
  - creates a list of elements for which a function returns true.
  - return an **filter object (iterator)**

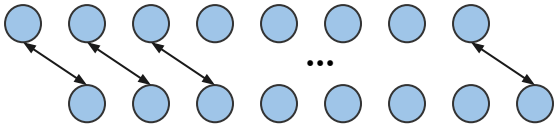



```
def evenfilter(val):
      return True if val % 2 == 0 else False

lst = [1,2,3,4,5]

filterobj = filter(evenfilter, lst)

# lst = list(filterobj)
for value in filterobj:
      print (value)
```

Lab 14 : filter

- Write a function called `gtRight(numbers)` that uses `filter` and `zip` to filter out elements that are greater than the element to their right. (exclude the last element)
  - Return value: the list
  - Requirement: use `filter, zip`

filter **list** **zip**

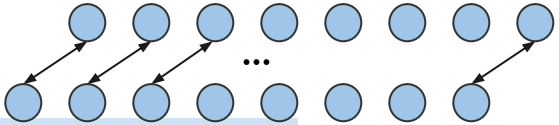



```
mylist = [ 5, 10, 15, 25, 20, 55, 40]

# Expected output
# return this list
[25, 55]
```

Lab 15 : filter

- Make a program using zip, filter and map
  - Write a function called `gtLeft(numbers)` that uses `filter` and `zip` to find elements in a list that are greater than the element to their left. Ignore the first element
  - Return value: the list
  - Requirement: use `filter, zip`

filter **list** **zip**

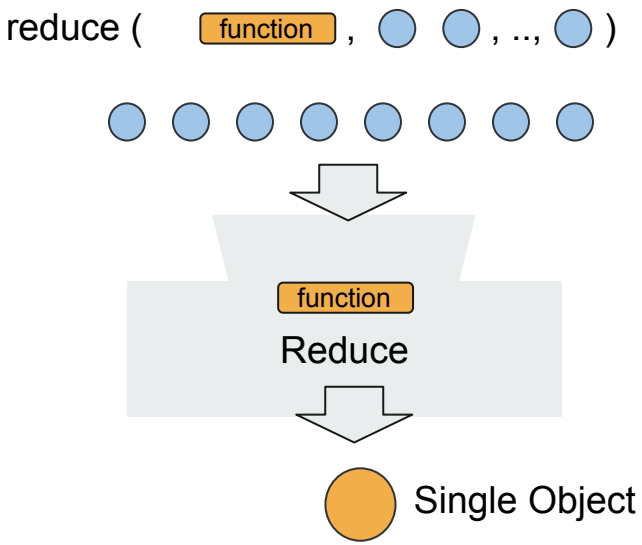



```
mylist = [ 5, 10, 15, 25, 20, 55, 40]

# Expected output
[10, 15, 25, 55]
```

reduce

Reduce

- Reduce
  - function for performing some computation on a iterable and
  - returns a single value

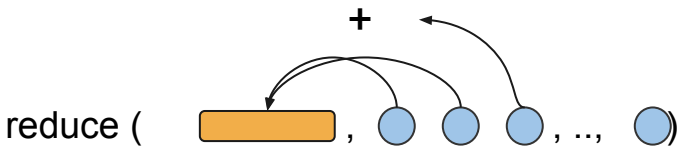



```
from functools import reduce

lst = [1,2,3,4,5,6,7,8,9,10]
# lst = [1,2,3,4, 5]

lstsum = lambda x, y: x + y

print (reduce(lstsum, lst))
```

Reduce

- Reduce
  - function for performing some computation on a iterable and
  - returns a single value

```
from functools import reduce

lst = [1,2,3,4,5,6,7,8,9,10]


lstsum = lambda x, y: x + y

print (reduce(lstsum, lst))
```



- **At first step, first two elements are calculated.**
- **Next, the result of the first step and the 3rd element are calculated**
- **And so on.**

Lab 16 : Reduce

- Write a function called `getMaxSum(numbers)` that takes a list of lists, `numbers`, and computes the sum of the maximum elements from each sublist in `numbers`."
- Requirement
  - Use `reduce()` function and create your lambda function to compute the max element summation

```
mylist = [[1, 2, 3, 4, 5],
          [10, 20, 30, 40, 50],
          [100, 200, 1000]
          ]

# Expected output
1055
```

```
####
def
numb
maxs
prin
# Re
```

Lab 17 : Reduce

- Write a function called `getAvg(numbers)` that takes a list of lists, `numbers`, and computes the average of all elements in `numbers`."
- Requirement
  - Use `reduce`( ) function and create your lambda function to compute the summation
- Make a program using reduce
  - get the average of the list

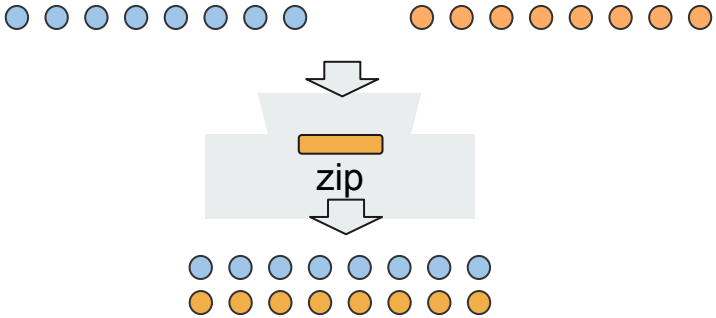
```
mylist = [ 5, 10, 15, 25, 20, 55, 40]

# Expected output
24
```

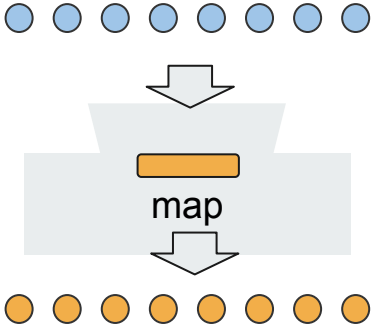
Reduce

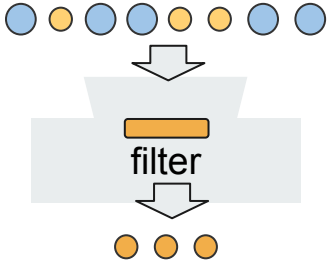
- Reduce
  - Using **operator**

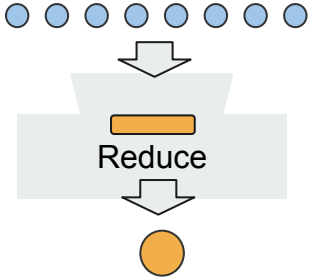
```
from functools import reduce
import operator

mylist = [1,2,3,4,5]

mysum = reduce(operator.add, mylist)
print (mysum)

productoflist = reduce(operator.mul, mylist)
print (productoflist)
```

Summary of zip, map, filter and reduce









Summary of zip, map, filter and reduce

Kyuwoong Lee, Ph. D

Assignments

Introduction to Python Programming

Instruction to submit your work

Make multiple commits, even if there are small changes.

It is strongly recommended to increase the **frequency of commit/push actions**

- When you accept the Classroom Assignment through the link, it is the beginning time of the question.
- After initializing the Repository, you should **at least commit/push every 5 minutes**.(frequent commits/push)
    - A major part has been built (e.g., for loop / if statement)
    - When you meet errors, try to fix errors,
- This will allow me to see your progress and how you have worked on your code
- This commit log can make us see the program's progress and prevent **plagiarism**.
    - **More commits/push, more points**
    - **Only one commit, no point.**

Assignment 1

	A	B	C	D	E
1	name	ID	Math	English	Computer
2	John Doe	2022-0002	100	90	90
3	Jane Doe	2023-0001	70	100	90
4	Mary Smith	2023-0002	100	100	100
5	Bill Watson	2023-0003	100	75	85

students.xlsx

- The purpose of Lab
- Step 1: Understanding the list of dictionary
  - Make a **list of dictionary**
- Step 2: Learn how to read lines from excel file
  - Learn DataFrame from ‘pandas’ to read the excel file
- Step 3: Learn how to construct the dictionary and append it to the list
  - The thinking way how to construct a list of dictionary from the excel file
- Step 4:
  - From the list of dictionary, find a particular value by the key

Assignment 1

- 10 mins to pandas
  - https://pandas.pydata.org/docs/user_guide/10min.html
- Dataframe from Pandas
  - Example Source Code
  - Click the link to see the source code and Run all code cells in this Jupyter notebook file.
    - Figure out how to access the row and column values in DataFrame

```
import pandas as pd


df = pd.read_excel('students.xlsx')
df
```

	name	ID	Math	English	Computer
0	John Doe	2022-0002	100	90	90
1	Jane Doe	2023-0001	70	100	90
2	Mary Smith	2023-0002	100	100	100
3	Bill Watson	2023-0003	100	75	85

```
df.columns      # heading
df.shape        # # of rows and columns
df.shape[0]     # # of rows
df.shape[1]     # # of columns

df.iloc[0]      # the first row
df['ID']        # column 'ID' values
```

Assignment 1

- (1) Create a function `makeStudentDictionary( )`
  - Parameter:
    - None
  - Task
    - Read a Excel file "students.xlsx" and make a "DataFrame"
      - Construct a dictionary from a "DataFrame".
      - See the Example Source.
  - **return value**
    - List of dictionaries for all students information (use the **same** key values)

```
[{'Name': 'John Doe', 'ID': '2022-0002', 'Math': 100, 'English': 90, 'Computer': 90},
{'Name': 'Jane Doe', 'ID': '2023-0001', 'Math': 70, 'English': 100, 'Computer': 90},
{'Name': 'Mary Smith', 'ID': '2023-0002', 'Math': 100, 'English': 100, 'Computer': 100},
{'Name': 'Bill Watson', 'ID': '2023-0003', 'Math': 100, 'English': 75, 'Computer': 85}]
```

```
name -> John Doe        ID -> 2022-0002        Math -> 100        English -> 90
name -> Jane Doe        ID -> 2023-0001        Math -> 70         English -> 100
name -> Mary Smith      ID -> 2023-0002        Math -> 100        English -> 100
name -> Bill Watson     ID -> 2023-0003        Math -> 100        English -> 75
```

Assignment 1

- (2) Create a function **`printStudents(list of dictionary)`**
  - Parameter
    - list of dictionary that was returned from the function **`makeStudentDictionary( )`**
  - Print all dictionary values with the user-friendly output format
  - return value
    - none

```
name -> John Doe        ID -> 2022-0002        Math -> 100        English -> 90
name -> Jane Doe        ID -> 2023-0001        Math -> 70         English -> 100
name -> Mary Smith      ID -> 2023-0002        Math -> 100        English -> 100
name -> Bill Watson     ID -> 2023-0003        Math -> 100        English -> 75
```

Example

Assignment 1

- (3) Create a function `findStudent(list of dict, int id)`
  - Parameters:
    - list of dictionaries
    - one integer value for ID
  - Find the students who have the given ID in the parameter
  - Return value
    - a list of scores of the student who has the same ID

```
[70, 100, 90]
```