COM3103

Artificial Intelligence

1. Python Programming – Data Structures

Lists

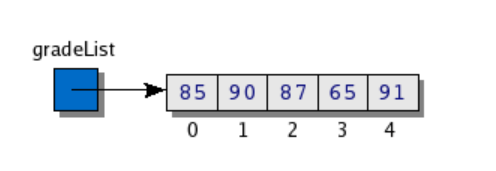
Lists in Python replace the function of *Arrays* and *ArrayLists* in Java.

// Java

int [] gradeList = { 85, 90, 87, 65, 91 };

//Python  
gradeList = [ 85, 90, 87, 65, 91 ]

List can be accessed using indexes just like in Strings. As usual, the first element is number 0.



What are the output of the following?

gradeList[0]

gradeList[5]

gradeList[2:3]

gradeList[ :-2]

List Length ( len( ) function )

E.g., x = len(gradeList)

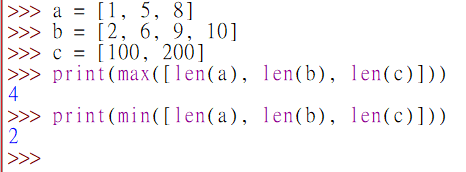
Note: len() also works for other data types, including string and set and dictionaries (see below)

Largest and smallest element ( max ( ) function, max ( ) function)

a = [1, 5, 8]

b = [2, 6, 9, 10]

c = [100, 200]

print(max([len(a), len(b), len(c)]))

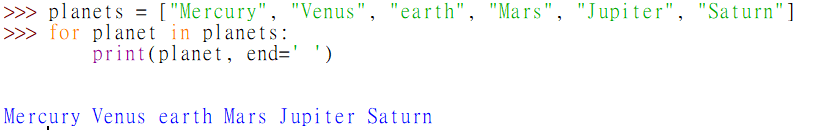
print(min([len(a), len(b), len(c)]))

Looping through a List

planets = ["Mercury", "Venus", "earth", "Mars", "Jupiter", "Saturn"]

for planet in planets:

print(planet, end=' ')



Note: end=' ' places a space at the end of the printed characters instead of a newline character.

Adding elements to the end of a list ( append() )

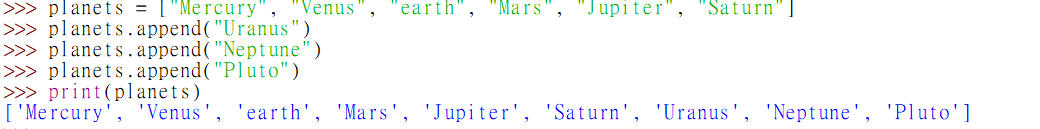
planets = ["Mercury", "Venus", "earth", "Mars", "Jupiter", "Saturn"]

planets.append("Uranus")

planets.append("Neptune")

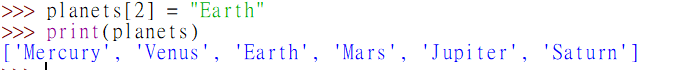
planets.append("Pluto")

print(planets)



planets = ["Mercury", "Venus", "earth", "Mars", "Jupiter", "Saturn"]

Changing an element in a List:



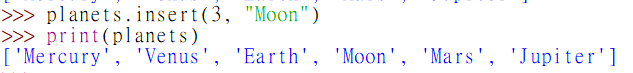
planets[2] = "Earth" print(planets)

Removing last element ( pop () )

planets.pop()

print(planets)

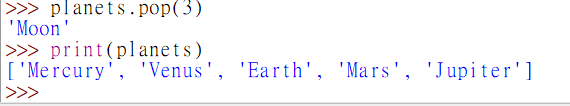
Inserting an element into given position (insert() )



planets.insert(3, "Moon")

print(planets)

Removing an element by number

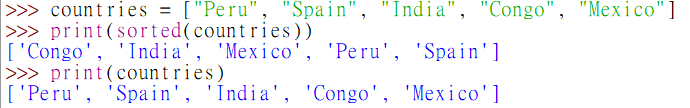
planets.pop(3)

print(planets)

# Moon is removed from the list

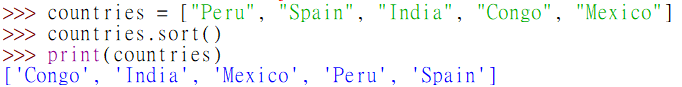
countries = ["Peru", "Spain", "India", "Congo", "Mexico"]

Returning a sorted version of a list (without changing the list itself):

   
print(sorted(countries))

print(countries)

Sorting a list (the list will be updated)

countries.sort()

print(countries)

Exercise 6: List

1. What is the output?

names = ["Carol", "Albert", "Ben", "Donna"]

names.append("Eugenia")

print(sorted(names))

1. Base on the following list definition

days\_in\_month = [31,28,31,30,31,30,31,31,30,31,30,31]

Write a program MonthDays.py to print out the number of days in any month as entered by the user. The following is a sample session



Exercise 6: List (continue)

1. Base on the following list definition

3

lucky\_dates = ['28 Jul 2020', '17 Jul 2021','26 Dec 2021', '11 Jan 2022', '6 Jun 2022','21 Jul 2022', '30 Nov 2022', '26 May 2023', '19 Sep 2023']

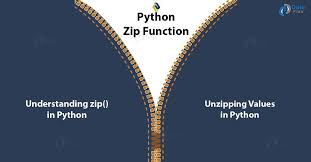
Write a program lucky.py to print out the dates of the *n* most recent lucky dates, where n is a number entered by the user. E.g.,

Enter the number of recent lucky you want to view: 3

['30 Nov 2022', '26 May 2023', '19 Sep 2023']

zip

Given two or more lists, e.g.**,**

months = ["Jan", "Feb", "Mar", "Apr", "May"]

days = [31, 28, 31, 30, 31]

months\_days = zip(months, days)

for (m, d) in months\_days:

print(m, d)

* months is a **list**: "Jan", "Feb", "Mar", "Apr", "May"
* days is a **list**: 31, 28, 31, 30, 31
* months\_days is a **zip object:** ("Jan", 31), ("Feb", 28), ("Mar", 31) …

zip

It is possible to convert a zip objects to other data structure, e.g.**, a dictionary**

months = ["Jan", "Feb", "Mar", "Apr", "May"]

days = [31, 28, 31, 30, 31]

months\_days = zip(months, days)

my\_dict = dict(months\_days)

print(my\_dict["Feb"]) #output 28

print(my\_dict["Apr"]) #output 30

map

To apply the same function to each element of a list.

E.g. given a list [3, 6, 9, 12, 15], let's product a list of the square root of each number:

import math

x = [3,6,9,12, 15]

sqrt\_x = **list(**map(math.sqrt, x)**)**

print (sqrt\_x)

In this example, we apply the sqrt function to each element in x.  
Note that we have converted the resulting map object to a list first before printint it:

Output:

Exercise 6: List (continue)

4 Given a list **x = [2, 3, 5, 7, 11,13]**, use **map** together with a **lambda function** to produce another new list **y**, where each element in **y** is the square of the corresponding element in **x.** Save your work as map\_lambda1.py

Tuples

Tuples are like lists except they are immutable (i.e, the **content cannot be added, removed, or changed**). For examples:

typhoon\_position = (20.2, 113.8)

print("The typhoon is located at ")

print("{} degree north ".format(typhoon\_position[0])

print("{} degree east ".format(typhoon\_position[1])

dimensions = 52, 40, 100 #the bracket can be omitted

length, width, height = dimensions #unpacking

print("The dimensions are {} x {} x {}".format(length, width, height))

Set

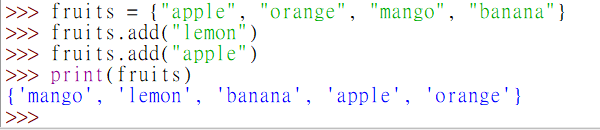
Sets are **unordered** collections of **unique** (i.e., **non-repeating**) elements.

fruits = {"apple", "orange", "mango", "banana"}

fruits.add("lemon")

fruits.add("apple")

print(fruit)



Note: the output is in an unpredictable order!

Example 6.1: counting unique words in a text

Write a program to count the number of unique words in this sentence.

*“Finished files are the result of years of scientific study combined with the experience of years.*

sentence = """finished files are the result of years of scientific study

combined with the experience of years"""

sentence\_set = set (sentence.split())

print (len(sentence\_set))

Dictionary

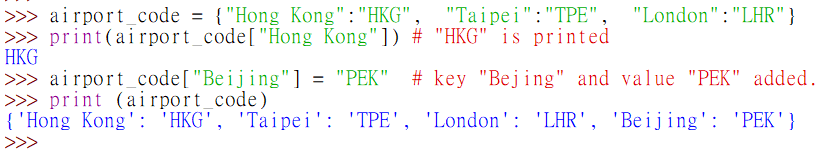
Dictionary stores pairs of keys and values.

The keys must be unique, and of immutable data type (e.g., number, string, tuple)

airport\_code = {"Hong Kong":"HKG", "Taipei":"TPE", "London":"LHR"}

print(airport\_code["Hong Kong"]) # "HKG" is printed

airport\_code["Beijing"] = "PEK" # key "Beijing" and value "PEK" added.

print (airport\_code)

Dictionary: Two methods for looking up values.

* using index (e.g., my\_dict["apple"])
  + result in error if the key does not exist.
* using the .get() method
  + no error if key does not exist, can supply default values if key not found

scores = {"Harry": 78, "Ron":63, "Ginny":80}   
scores["Harry"] #OK  
scores.get("Harry") #OK, same as above  
scores["Malfoy"] #ERROR: not found

scores.get("Malfoy") #None is returned, No error.

scores.get("malfoy",0) 0 is returned.

Exercise 7.1: Dictionary   
Create a dictionary for storing the following data

|  |  |
| --- | --- |
| **Country/Region** | **GDP (per capita)** |
| Luxembourg | 135046 |
| Norway | 99481 |
| Hong Kong | 49850 |
| UK | 49761 |
| China | 14096 |
| Brazil | 8570 |
| Kenya | 2252 |

1. Write a program (**gdp.py**) that allows the use to enter the name of a country or region, and print out its *GDP (per capita)* accordingly. Print out an appropriate message if the data does not exist in the table above. The program will repeat until the user enters the “exit” command.

Text

Description automatically generated

Data source: https://en.wikipedia.org/wiki/List\_of\_countries\_by\_GDP\_(nominal)\_per\_capita

Exercise 7.2: Dictionary

The value of a dictionary can be a data structure, for example, a list, a set or another dictionary.

animals = {'dogs': [20, 10, 15, 8], 'cats': [3,4,2], 'rabbits': [2, 3, 3]}

What is the value of?

1. animals['dogs']
2. animals['dogs'][3]
3. animals[2]
4. animals['elephant']
5. animals.get('elephant')
6. animals.get('cats')[1]

Example: Word frequencies

Count the number of appearance of each word

*“finished files are the result of years of scientific*

*study combined with the experience of years”*

**#Version 1**

sentence = """finished files are the

result of years of scientific study

combined with the experience of years"""

words\_freq ={} #dictionary for the counts

**for word in sentence.split():**

**if word not in words\_freq:**

**words\_freq[word] =1**

**else:**

**words\_freq[word] +=1**

print (words\_freq)

Example: Word frequencies

Count the number of appearance of each word

*“finished files are the result of years of scientific study combined with the   
 experience of years”*

**#Version 2**

sentence = """finished files are the result of years of scientific study combined with the experience of years"""

words\_freq = {}

for word in sentence.split():

words\_freq[word] = words\_freq.get(word, 0) +1

print (words\_freq)

Example: Looping through a dictionary

items = {"apple":3, "banana": 4, "pen": 3, "orange":2, "cup":1}

fruits = {"apple", "banana", "orange", "mango", "lemon"}

Count the total number of fruit items in the dictionary

fruit\_count, non\_fruit\_count = 0,0  
for i in items:

if i in fruits:

fruit\_count += items[i]

else:

non\_fruit\_count += items[i]

print(fruit\_count, non\_fruit\_count)

List Comprehensions

Example: Suppose we want to create a list of the squares of all numbers between 1 and 9.

my\_list = [] #FOR LOOP VERSION

for i in range(1,10):

 my\_list.append(i\*i)

print(my\_list)

We can simply this using Python's *List Comprehensions* Features

**my\_list = [ i\*i for i in range(1, 10)]**

print(my\_list)

List Comprehensions

Example: Create a list of the squares of all *even* numbers between 1 and 9.

my\_list = [ i\*i for i in range(1, 10) **if i % 2 ==0**]

print(my\_list)

Example: Given the following list

my\_list = ["UNO", "DOS", "TRES", "CUATRO", "CINCO"]

Create another list that is same as my\_list in content, except that all words are in lower case.

my\_list2 = [word.lower() for word in my\_list ]



Exercise 8: List Comprehension

**8.1** Given the following list of names.

names = ["Albert Yip", "Simon Pao", "Joe Sun", "Paul Choi"]

Use list comprehension to create and print out a new list containing the first names only

**8.2** Use list comprehension to create and print out a list containing the first 20 multiples of 3

**8.3** Given the following dictionary of student names and scores

scores = {"Anna":80, "Ben":30, "Cindy":72, "Don":20, "Edmond": 100}

Use list comprehension to create and print out a new list containing the names of the people who has passed (the passing mark is 40)

8.4 Write a program ***Actor*.py** for the following problem.

Given the following data.

*Nominations for the Hong Kong Film Award - Best Actor Award from 2013 – 2020:*

2013: "Tony Leung Ka-fai", "Nick Cheung", "Chapman To", "Sean Lau", "Tony Leung Chiu-Wai"

2014: "Nick Cheung", "Tony Leung Chiu-Wai", "Louis Koo", "Sean Lau", "Anthony Wong"

2015: "Sean Lau", "Eddie Peng", "Sean Lau", "Huang Bo", "Daniel Wu"

2016: "Aaron Kwok", "Andy Lau", "Nick Cheung", "Tony Leung Ka-fai", "Jacky Cheung"

2017: "Gordon Lam", "Shawn Yue", "Francis Ng", "Richie Jen", "Tony Leung Ka-fai"

2018: "Ronald Cheng", "Andy Lau", "Tian Zhuangzhuang", "Ling Man-lung"

2019: "Anthony Wong", "Francis Ng", "Chow Yun-fat", "Aaron Kwok", "Philip Keung"  
2020: "Tai Bo", "Louis Koo", "Chu Pak Hong", "Aaron Kwok", "Jackson Yee"

2022: "Gordon Lam", "Gordon Lam", "Leung Chung-hang", "Francis Ng"

1. Store the above data in a dictionary, similar to the one illustrated below:
2. Text, letter

   Description automatically generatedText, letter

   Description automatically generatedWrite a program to compute the number of nonimations won by each actor. Store the result in dictionary format, and print out the result.

*Sample output:*

Hints: The *word frequencies* example may be helpful for the counting part. The difference is that in the word frequecies exercise, the words are in a single list. In this example, the actors names are in separate lists that are the values of a dictionary (the dictionary keys are the years). How can you loop through each list in turn? (hints: use a double for loop). Store the results in another dictionary.

Then you can use another loop to print out the entries in the dictionary for the result.

C.

Continuing from the above, find out which actors has won the most nominations. Store the result in list format. Print out the result. (A list format is chosen as the output format because there may be more than one actor with the highest number of nominations)

  
Sample output:

**8.5** Write a program **actress.py** for the following problem.

Text

Description automatically generatedGiven a data file (actress.json) in JSON format that contains the years and names of the actresses nominated for the Best Actress Award in HK between 2013 and 2019:

Write a program toChart

Description automatically generated with medium confidence show a report of how many nominations have been won by each actress from 2013 to 2019. For example, part of the reports may look like:

Part of the program has been provided for you (especially the part for reading the data file).

Find out which actress(es) has won the most nomination. For example, the output may look like.

**Hint:**

* Place the data file in the same folder as your program.
* Inside the program, the year and the list of actress is stored into a dictionary called nominated.
* For example, after reading the data, the line

names = nominated.get("2015")

would give you the list of actress in 2015 (and assign them to the variable *names*).

* You may consider to use another dictionary to store the number of occurrence of each name.
* You may need to loop through each entry in nominated, and update the counts in your dictionary you defined in step 3.

Example: ***The WOLF, SHEEP and CABBAGE Puzzle***

<https://www.proprofs.com/games/wolf-sheep-and-cabbage/>

* See the sample program on Moodle

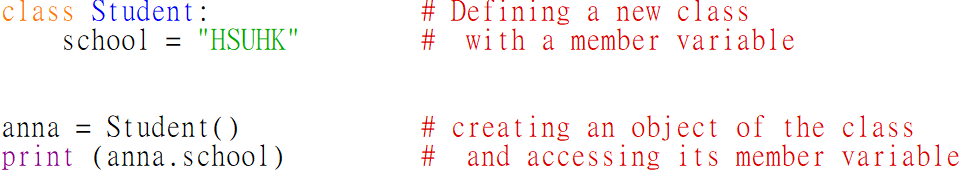
Diagram

Description automatically generated

Class and Objects in Python (Reference)

Just like other object oriented programming language, Python has language features for defining classes, methods and objects.

**A simple class in Python**

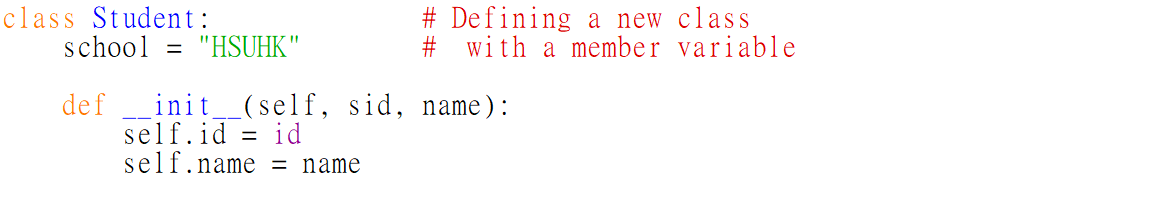
****

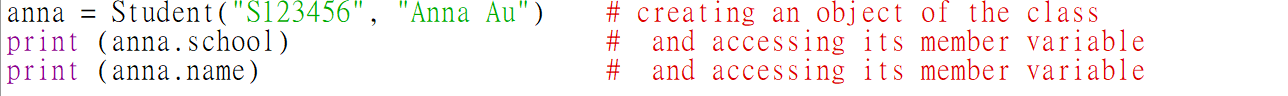
****Sample output:

**The \_\_init\_\_() function**

The \_\_init\_\_() function is called automatically when an object is created. (Like the Java constructor)

For example, the following code will set up a Student object with a sid and a name:





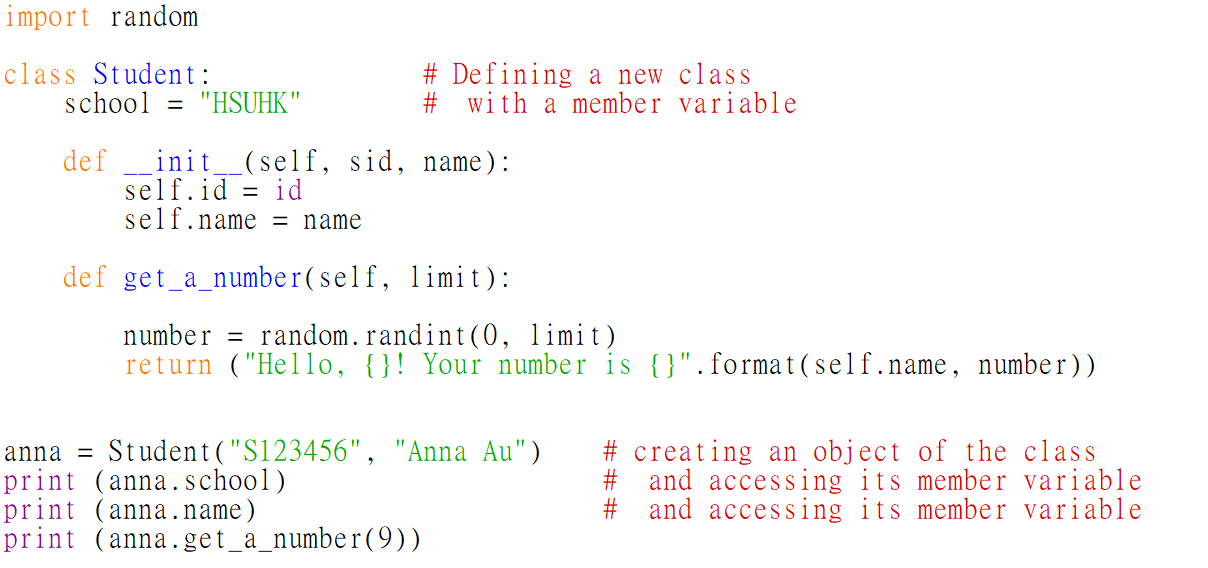


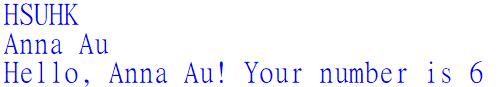
Sample output:

Note: the **self** parameter is a reference to the current instance of the class, which must be the first parameter of any method definition.

**Methods, Parameters and Return**

Defining Methods is straightforward. For example, the following code defines a method called *get\_a\_number()*, which takes a parameter *limit*, and returns a greeting message together with a random lucky number between 0 and *limit* to the caller.



Sample output: