Beihang University — Beijing, Oct. 18th, 2019
School of General Engineering — Sebastian Wandelt, Pengfei Yi, Junyu Zhao

Computer Science and Programming
Lab Class 5

Task 1. ***Recursion – Review*** *(15 minutes)*

In the last two weeks, you are dealing with the prime numbers with loops, and this task is asking you to write a function that verifies if one given number is a prime number or not in a **recursive** way.

Hint: To achieve that, you should consider changing parameter values in each recursive calls. To do so conveniently, you might want to define the default parameter values when defining the function. As shown in the code below,

```
def addition(a=1,b=1):
    print (a+b)
addition(2,4)
>>>6
addition(2)
>>>3
addition()
>>>2
```

The function `addition` needs two parameters `a` and `b` and they both have the default value of 1, which means if one or both of their values are not given when calling the function, the value of the non-defined parameter will automatically be 1. When calling the function with only one value, `addition(2)`, the value of `a` is 2, defined by the user and the value of `b` is 1 by default.

Task 2. ***Class – Rectangle*** *(15 minutes)*

Please implement a class for calculating the meaningful parameters of rectangles named `Rectangle`. The class should have two input values, the rectangle's `length` and `width`.

(1) Define the *constructor function* (if you forget how to do it, go back to the lecture slides) and the *instance variables* with the two input values.

(2) Define one rectangle, and print out its instance variables' values outside the class.

(3) Define a function in the class that *prints out the instance variables*, call it to see if it works as you predicted. Compare the two ways of showing the instance variables' values in (2) and (3), which one is better? Explain.

(4) Define a function in the class that *calculates the area of the rectangle*. Define one rectangle and call the function to see if it works as you predicted.

(5) Define a function in the class that *compares the areas between two rectangles*. Think about how should the input of this function be defined. Call it once to see if it works as predicted. For finding the correct member function name, please read the Python documentation.

Task 3. *Class – Calender* *(30 minutes)*

Suppose you want to build a calender which records the recent events that you arranged for yourself. For instance, you might asked someone out for a movie this Sunday morning and you need to attend a Boya lecture held by SGE in the evening of next Monday. What you need to do is to build a class that is for one event and then build a calender class that uses the construction of the event class and sort the events properly.

In specific, you could follow the sub-tasks step by step to achieve the goal.

(1) Define a class for **one event** named `Event`. Considering the important information that should be recorded in the calender of one event, the class should have the event's name, time, location as its input. Define the *constructor function* and the *instance variables*. Note that you should make those information callable for they might be useful in making the calender class. An example of using the class is shown below, where the instance variables are called `name`, `time` and `location`.

```
BoyaLecuture=Event('boya lecture about research','Morning of
                            Oct 20','D1120')
print (BoyaLecture.name)
>>>'boya lecture about research'
print (BoyaLecture.time)
>>>'Morning of Oct 20'
print (BoyaLecture.location)
>>>'D1120'
```

(2) For more conveniently using the information of an object in the class `Event`, you should *formalize the input information*. That is, for each of the information, the user should know how should it be formalized to. For instance, you could tell the user that the date of the event should be in form of `'10/20'` rather than `'Oct, 20th'` because the first form is easier for further operations.

An intuitive way is to separate the time information into the date and the exact time in that day. That is, for an input like `'10-20, Morning'`, it should be recorded with `'10-20'` and `'Morning'`. Therefore, you should modify the class `Event` to make it has several formalized instance variables.

```
BoyaLecuture=Event('boya lecture about research','10-20','
                            Morning','D1120')
print (BoyaLecture.name)
>>>'boya lecture about research'
print (BoyaLecture.date)
```

100

140

90

255

50

285

```
>>>'10-20'
print (BoyaLecture.time)
>>>'Morning'
print (BoyaLecture.location)
>>>'D1120'
```

(3) Define the class for **the calender** named `Calender`. Define its constructor and a `NewEvent` function that adds one event to the calender. For the beginning, the class `Calender` could simply use a list to record all its events. Add two events to the your calender and see if it works as you predicted. What do you see when simply print out the list with events? How could you solve the problem and make the events visible?

```
BoyaLecuture=Event('boya lecture about research','10-20','
                              Morning','D1120')
ICSPlab=Event('Lab class', '10-18', 'Morning','D1120')

MyCalender=Calender()
MyCalender.NewEvent(BoyaLecuture)
MyCalender.NewEvent(ICSPlab)
```

(4) To avoid time conflicts, the `Calender` class should have one function that is for *checking time availability* when one new event is added. Build a function that when a time conflict happens, print out that the new event couldn't be added and stop executing the `NewEvent` function.

(5) Build a function that *print out all the events' name on one day* with a given day as its input named `findEvents`, as shown in the code below.

```
MyCalender=Calender()
...#The additions of events
MyCalender.findEvents('10-20')
>>>'boya lecture about research'
```

Task 4. ***Class – Matrix*** *(30 minutes)*

Write the code for computing the row reduced echelon form of a matrix from the Linear Algebra lecture.