Beihang University
School of General Engineering

Beijing, October 27th, 2020

Lin Wei and Yida Ding

Computer Science and Programming Lab Class 5

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

(1) 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.

(2)Build a dictionary in Python which stores for each number in the interval [2,1000], whether it is a prime number. If the number (=key) is a prime number, then the value of the dictionary element should be ‘yes’. Otherwise, the value should be ‘no’

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 – Pet dog*** *(25 minutes)*

Implement a Python class `Dog`, which satisfies the following additional constraints:

1. There should be an initial function which generates a dog that has not eaten anything and has not been walked for any time.
2. It can be fed by a function `feed()` and the value of `eaten_food` will be added by one.
3. It can be walked by a function `walk()` and the value of `walk_time` will be added by one.
4. Its mood can be checked by a function `check_mood()`. If `eaten_food<3` or `walk_time<1`, the mood is `angry`; otherwise, the mood is `happy`.
5. Everyday, all states (`eaten_food` and `walk_time`) of this dog will be reset to zero.

Task 4 ***Class – Pet dog (further)*** *(40 minutes)*

Further modify the class `Dog` in the previous task, which satisfies the following constraints:

1. Add a new state `sex` $\in \{male, female\}$ to the class.
2. Two dogs could meet each other by function `meet()`. If a male dog and a female dog meet, both of them will have `happy` mood; If two dogs with the same `sex` meet and the `eaten_food` values are larger than zero for both of them, they will `fight` with each other, the `angry` dog will win. If both dogs have the same mood, the dog with larger `eaten_food` will win. If both dogs have the same mood and equal `eaten_food`, both of them will fail.
3. After the fight, winning dogs will be `happy` and the dogs that lose will be `angry`. In addition, the values of `eaten_food` will be reduced by one for both dogs.

Only once you are finished with all tasks above:

Task 5 ***Class – Matrix***

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