Computation Experiments 實驗15-16

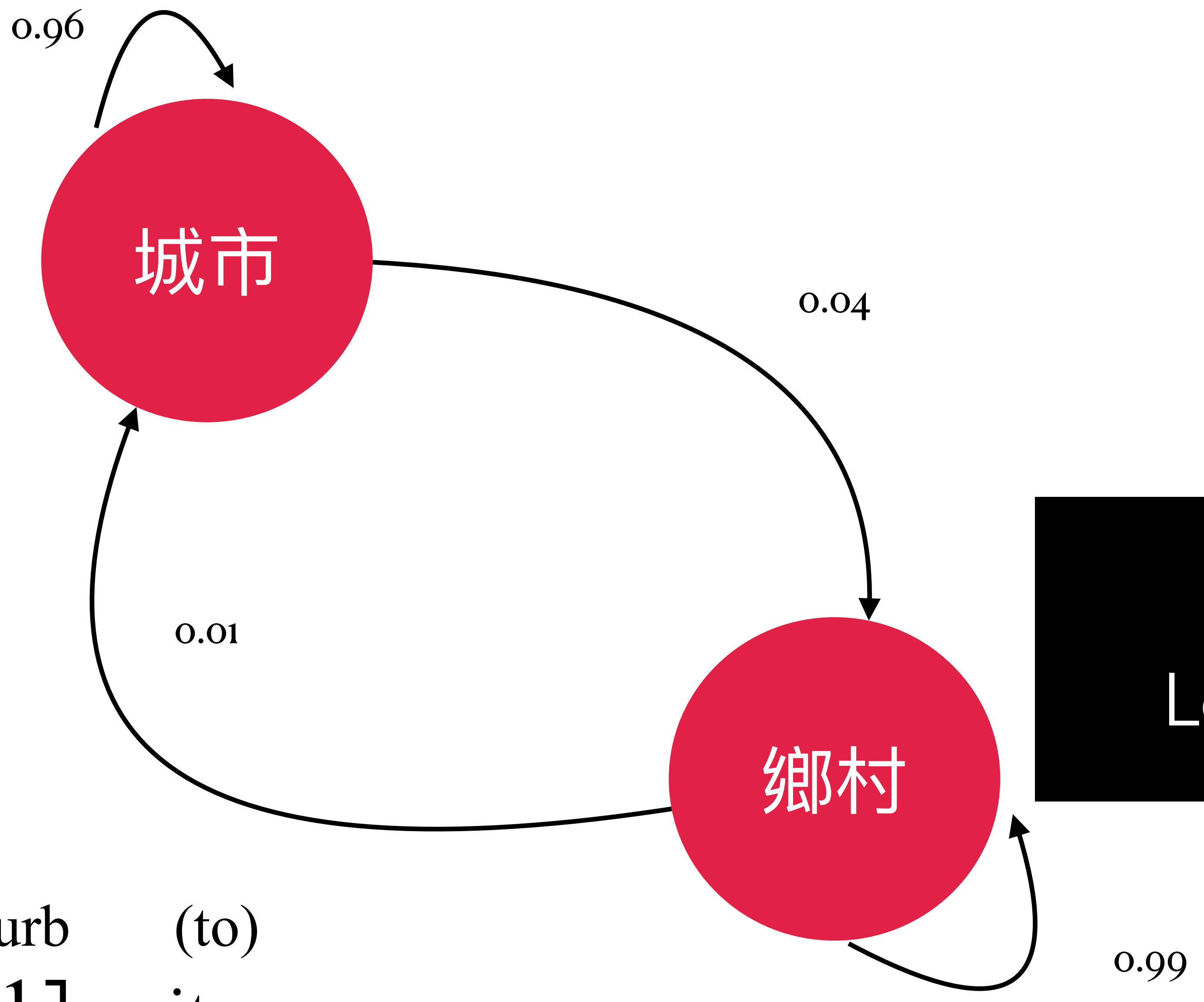
- 物件導向：類別Square、RegularTriangle、Stack堆疊設計
- 遞迴程式設計 n!

Ex15A 模擬馬可夫鏈狀態轉移

定義狀態轉移模擬模組sim

1. 新增markov.py
2. 匯入numpy
3. 函數輸入A 代表轉移矩陣，函數x代表初始狀態，定義方法sim(A, x)
4. 使用while迴圈模擬狀態轉移
   - 設計迴圈進入條件
   - 更新狀態
   - 超過最大迴圈數，break

馬可夫鏈的移轉機率
Markov chain



Refer to Lecture12

$$P = \begin{array}{ccc} \text{city} & \text{suburb} & \\ \left[\begin{matrix} 0.96 & 0.01 \\ 0.04 & 0.99 \end{matrix}\right. & \left.\right] & \begin{matrix} \text{city} \\ \text{suburb} \end{matrix} \end{array}$$

(from) city suburb (to)

Markov chain
初始人口數設定

```
import numpy as np
from numpy.linalg import no

x = np.array([[80],[175]])
print(norm(x))
print(np.sqrt(x[0]*x[0]+x[1]*x[1]))
```

$$\|x\| = \sqrt{x_1^2 + x_2^2}$$

```
192.41881404893857
[192.41881405]
```

Ex15B. Define class NamedShape

Define class NamedShape

Step 1. definition

- Declare variables numberOfSides and name
- Use __init__ to set variable name to an input argument
- Define a method for simple description

Refer toSlide page 6 of Lecture 13

宣告稱為NamedShape的類別

幻燈片子標題

兩個內建特性（變數）

```
class NamedShape:
    numberOfSides = 0
    name = ""
    def __init__(self,name):
        self.name = name

    def simpleDecription(self):
        ss = "A shape with sides of "+str(self.numberOfSides)
        return ss
```

類別本身的欄位變數或特性變數

Python內建的初始化方法，傳入輸入參數name將內建變數name的內容設定為輸入參數name的內容

Refer toSlide page 6

Step 2. Declare a variable of class NamedShape

- Declare namedShape as a variable of class NamedShape
- Set the builtin variable numberOfSides of namedShape to 7
- Print simple description of variable namedShape

Ex15C Define class Square

class Square(NamedShape)

Step1. Class definition

- Define class Square
- The parent of class Square is set to class NamedShape
- Declare variable sideLength
- Use __init__(self, sideLength) to define the initialization method
  - Set variable sideLength
  - Use super().init(name) to set variable name, which is inherited from class NamedShape
  - Set variable sideOfNumber to 4
- Define method area
- Define overwrite method simpleDescription

Refer to Slide page 9 of Lecture 13

宣告類別為NamedShape的類別Square

類別Square繼承母類別NamedShape特性與方法

```python
class Square(NamedShape):
    sideLength = 0.0
    def __init__(self,sideLength,name):
        self.sideLength = sideLength
        super().__init__(name)
        self.numberOfSides = 4
    def area(self):
        return self.sideLength * self.sideLength

square = Square(2.0, "mySquare")
print("name: ",square.name)
print("area: ", square.area())
```

Public variable 內建特性

使用母類別中的init函數設定母類別的變數name

定義方法area

宣告變數square的類別為Square，設定兩個輸入參數
印出square變數的名稱與面積

```
name:   mySquare
area:   4.0
```

9

Refer to Slide page 9 of Lecture 13

Step 2. Declare a variable of class Square

- Declare square as a variable of class Square
- Print the builtin variable name of square
- Print the simple description of square

Ex15D. Define class RegularTriangle

Define class regularTriangle

Step1. Definition

- Define class RegularTriangle(NamedShape)
- Declare variable sideLength
- Use __init__(self, name) to define the initialization method
    - Use super().__init__ to set variable name
    - Set variable numberOfSide
- Define method get_perimeter(self)
    - Return perimeter
- Define method set_perimeter(self, newValue)
    - Use newValue to set variable sideLength
- Define an overwrite method simpleDescription(self)

Refer to Slide page 14 of Lecture 13

Step 2. Declare a variable of class RegularTriangle

- Declare triangle as a variable of class RegularTriangle
- Print the result of triangle.get_perimeter()
- Set the perimeter of triangle to 9.9
- Print the result of triangle.get_perimeter()

Ex15E. Define class Complex

class Complex

Step.1 Definition of class Complex

- Define class Complex
- Use __init__(self, realPart, imagePart) to define the initialization method
  - In the method, set variable r to the input argument realPart
  - In the method, set variable I to the input argument imagePart
- Define method addComplex(self, d)
  - Add d.r to self.r
  - Add d.i to self.i
- Define method multiplyComplex(self, f)
  - Update self.r and self.i to the result of multiplying complex f to self

Refer toSlide page 21 of Lecture 13

Step 2. Declare variables of class Complex

- Declare c as a variable of class Complex for denoting $3 - 4.5i$
- Declare d as a variable of class Complex for denoting $2 + 1.5i$
- Print the result of c.addComplex(d)
- Print the result of c.multiplyComplex(d)

EX16A. 設計攝氏溫度類別，具備轉換華氏溫度的功能

Refer to slides 2-5 of Lecture 14

- 定義**class** Celsius
- 定義初始化方法，使用set_temperature 設定雙底線變數__temperature的內容
- 定義set_temperature， 設定雙底線變數__temperature的內容
- 定義get_temperature，回傳雙底線變數__temperature的內容
- 定義to_fahrenheit，回傳華氏溫度

- 宣告human 為類別Celsius變數，溫度值為36.0
- 使用get_temperature印出human的溫度
- 使用to_fahrenheit，印出human的華氏溫度

Ex16B. (25 points) 物件導向：Stack Design 堆疊設計

1. 宣告類別Stack
2. 設計初始化方法
3. 設計is_empty方法
4. 設計push方法
5. 設計pop方法

Define class Stack

Step 1. Definition

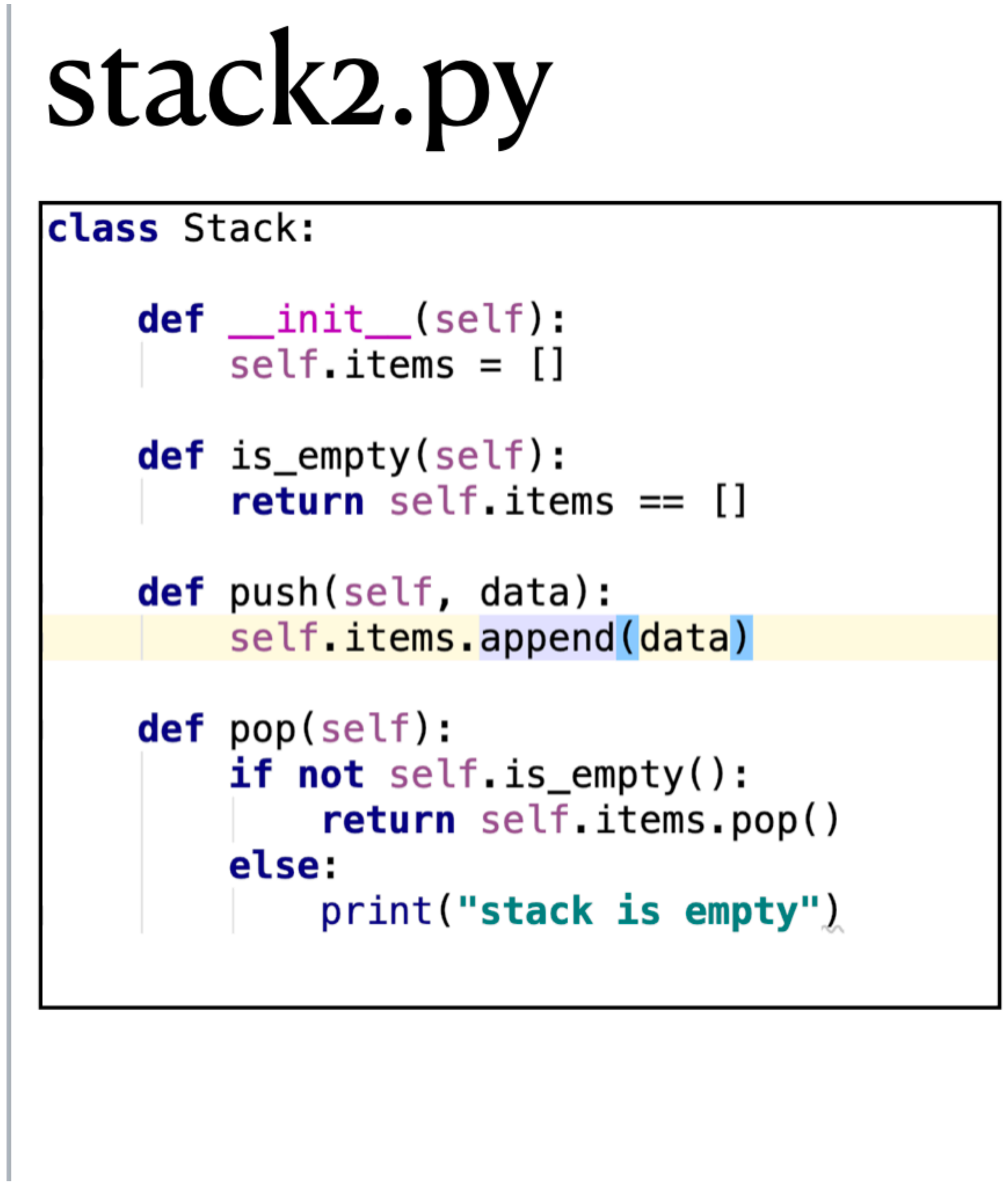
- Declare class Stack
- Use __init__(self) to define the initialization method
    - Set the builtin variable items to an empty list []
- Define the method is_empty
    - Return true if self.items is empty and false otherwise
- Define the method push(self, data)
    - Push data to the stack
- Define the method pop(self)
    - Return the item popped from the stack

Step 2.

- Declare s as a variable of class Stack
- Push “123” to stack s
- Push “2” to stack s
- Push “*” to stack s
- Set variable op to the result popped from stack s and print op
- Set data1 to the result popped from stack s and print data1
- Set data2 to the result popped from stack s and print data2

參考範例

slides 11-18 of Lecture 15



stack2.py

```python
class Stack:

    def __init__(self):
        self.items = []

    def is_empty(self):
        return self.items == []

    def push(self, data):
        self.items.append(data)

    def pop(self):
        if not self.is_empty():
            return self.items.pop()
        else:
            print("stack is empty")
```

EX16C. (25 points) Recursive programming 遞迴程式設計 n!

參考內容slides
4-6,20-22

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n(n-1)! & \text{if } n > 0 \end{cases} \qquad \forall n \in \mathbb{N}.$$

步驟 step

1. 以遞迴程式設計，定義方法fac，使得輸入為n，回傳輸出為n!
2. 設定n 為12
3. 呼叫方法fac，將n傳入fac，印出回傳的答案

Steps

1. Define method fac(n) by recursive programming, where the method returns $n!$ and n denotes a non-negative integer
2. Set variable n to 12
3. Print the result of fac(n)