CS1010S Tutorial 10

Wang Zexin

National University of Singapore
Year 2 QF

wang.zexin@u.nus.edu

April 10, 2017

Today's Agenda

1. Recap
2. Question One
3. Question 2
4. Extra stuff: Two simple final questions
    - A simple final question on Message Parsing
    - A simple final question on OOP

Recap - OOP

- What are classes and objects?
- Blueprints/models
- Initialization
  - $\_\_init\_\_$
  - $self.what = what$
- Encapsulation
  - Your attributes should invisible.
  - Use secret attribute names.
  - You should be assigning getters for those which should be public.
  - $self.\_\_name$ is somehow ‘secret’
  - $getName()$ method is public!
- Inheritance
  - No need to redefine attributes and methods for subclasses.
  - Format: $classA(B):$
  - Call superclass′s method: $super.()\_\_init\_\_()$

Recap - OOP

- Multiple Inheritance
  - Format: $classA(B, C):$
  - All the attributes and methods inherited
  - What if there are two different methods in both superclasses?
  - Use B
  - Which one is called when you call for $super()$?
  - B
- Polymorphism through Inheritance!
  - Different methods with the same name
  - Overriding methods: rewrite methods
  - Overriding attributes: rewrite constructor
  - You can add more content after calling the superclass method!
  - Overriding operators: rewrite $\_\_eq\_\_$ / $\_\_add\_\_$
- To check the attributes/methods $dir(objectName)$
- To check the class of object: $isinstance(objectName, className)$

Recap - Momoization

A method to reduce computation costs of your recursive function!

- Rationale: originally your recursion has a very long runtime.
- We want to reduce the number of recursions done!
- Suppose your function is f, we will reduce the number of recursions
- by recording down the already computed values of $f(x)$.
- A simple recipe
    - Create a table to record all the values of $f(x)$
    - Everytime when you call $f(x)$,
    - check if the value already exists or not.
    - If exists, return the value.
    - Otherwise compute $f(x)$.
- A common trick is to use a Dictionary to implement this table
- Use x as the key and $f(x)$ as the value.
- Question: what if there are multiple arguments?

Recap - Dynamic Programming

A method to change from top-down to bottom-up in computations!

- i.e. we can change recursion into iteration!
- At a cost of the storage space
- Note that we are effectively trading space for time here.
- There is no fixed method to do DP.
- We simply give a sketch of the method here.
- A simple recipe
  - Come up with a recurrence relation equation to
  - Calculate the higher level values with lower level ones
  - For example, $f(x) = f(x-1) + f(g(x))$
  - We then attempt to reverse or restructure g.
  - To obtain this: $f(x) = h(f(x-1), f(x-2), \ldots, f(1))$
  - Iterations can be carried out afterwards.

Recap - Dynamic Programming

We shall go through a demo on DP for prime numbers.
Problem is : how do we find all the prime numbers not larger than N?

- DP for prime numbers has the name Sieve Method.
- Instead of checking whether each number has a factor or not,
- we build a list with prime numbers and potential prime numbers.
- For any numbers in the prime number list,
- we delete their multiples from the potential list.
- When we finish checking the prime number list,
- the smallest potential prime number becomes prime number.

Recap - Exception Handling

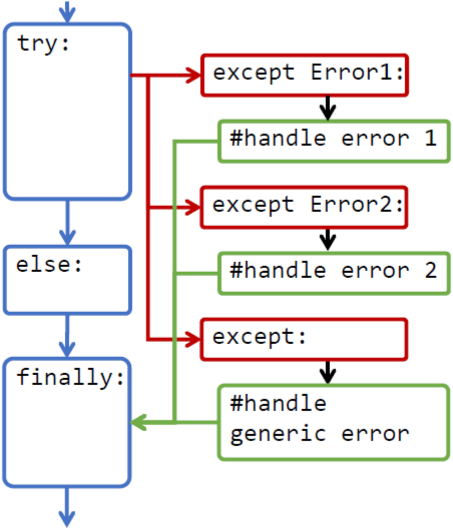
Rationale: we prefer to see outputs instead of errors.

- Syntax for raising an error: raise NameError('HiThere')
- Python built-in exceptions: http://docs.python.org/3.3/library/exceptions.htm#bltin-exceptions
- try:
- do something
- except SomeError:
- print/throw something
- else:
- print('no error!')
- finally:
- print('some other error!')

Recap - Exception Handling

When never you forget the syntax of error handling, refer to this diagram.

```
try:
    # statements
except Error1:
    # handle error 1
except Error2:
    # handle error 2
except: # wildcard
    # handle generic error
else:
    # no error raised
finally:
    # always executed
```



Question 1 Part A

Write a function *collatz_distance*(*n*).

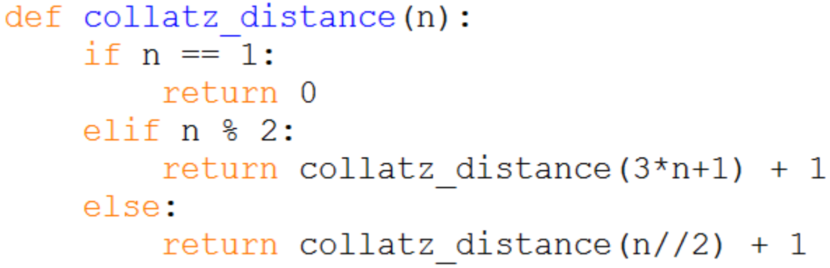
- The distance to go from n to 1.
- If odd, multiple by 3 and add 1.
- If even, divide by 2.

Question 1 Part A Discussion

We shall just write a simple recursive function.

- Base case: if $n = 1$, the number of steps should be 0.
- If odd, the number of steps to go from $3n + 1$ to reach 1 plus 1.
- If even, the number of steps to go from $\frac{n}{2}$ to reach 1 plus 1

Question 1 Part A Zexin's solution



```
def collatz_distance(n):
    if n == 1:
        return 0
    elif n % 2:
        return collatz_distance(3*n+1) + 1
    else:
        return collatz_distance(n//2) + 1
```

Question 1 Part B

Write a function *max_collatz_distance*(n).

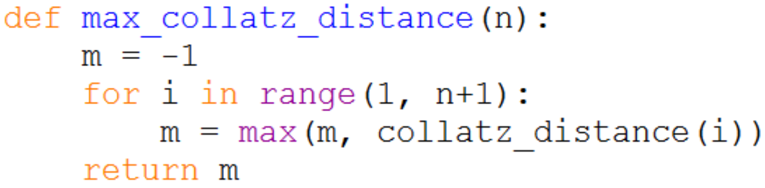
- This function computes the maximum value
- of *collatz_distance*(1), *collatz_distance*(2), . . . , *collatz_distance*(n).

Question 1 Part B Discussion

We shall just write a simple iterative function.

- Set the maximum value to 0.
- Loop from 1 to n.
- If greater than maximum, replace the maximum.

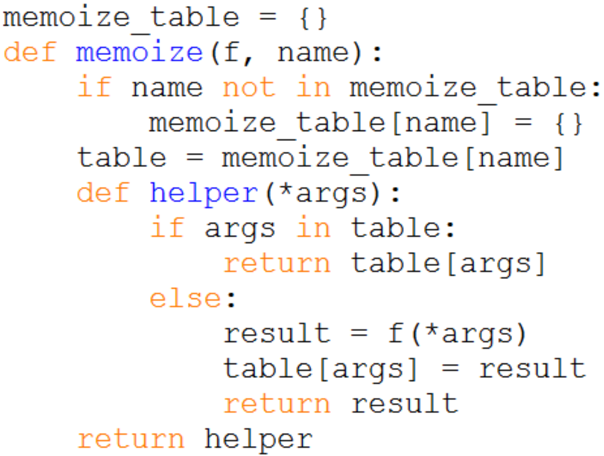
Question 1 Part B Zexin's solution



```python
def max_collatz_distance(n):
    m = -1
    for i in range(1, n+1):
        m = max(m, collatz_distance(i))
    return m
```

Question 1 Part C

Give a memoized version of *max_collatz_distance_memo*(*n*) using memoize as provided in the lecture.



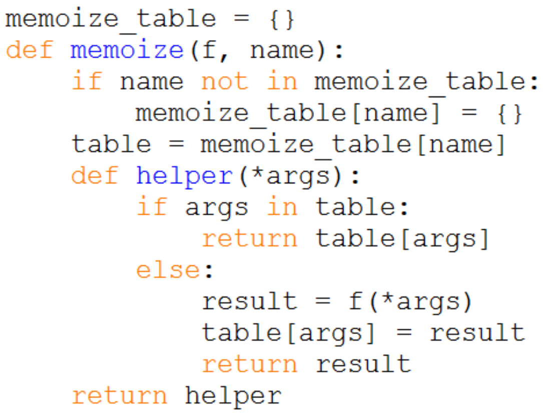
```
memoize_table = {}
def memoize(f, name):
    if name not in memoize_table:
        memoize_table[name] = {}
    table = memoize_table[name]
    def helper(*args):
        if args in table:
            return table[args]
        else:
            result = f(*args)
            table[args] = result
            return result
    return helper
```

Question 1 Part C Discussion

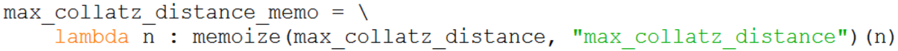
We will only need to write one line using this function.

- This function simply make use of a table to do memoization.
- What you need to input are just the function and a name.
- Function is just the *max_collatz_distance*.
- Does the name matter?

Question 1 Part C Zexin's solution



```
memoize_table = {}
def memoize(f, name):
    if name not in memoize_table:
        memoize_table[name] = {}
    table = memoize_table[name]
    def helper(*args):
        if args in table:
            return table[args]
        else:
            result = f(*args)
            table[args] = result
            return result
    return helper
```



```
max_collatz_distance_memo = \
    lambda n : memoize(max_collatz_distance, "max_collatz_distance")(n)
```

Question 1 Part D

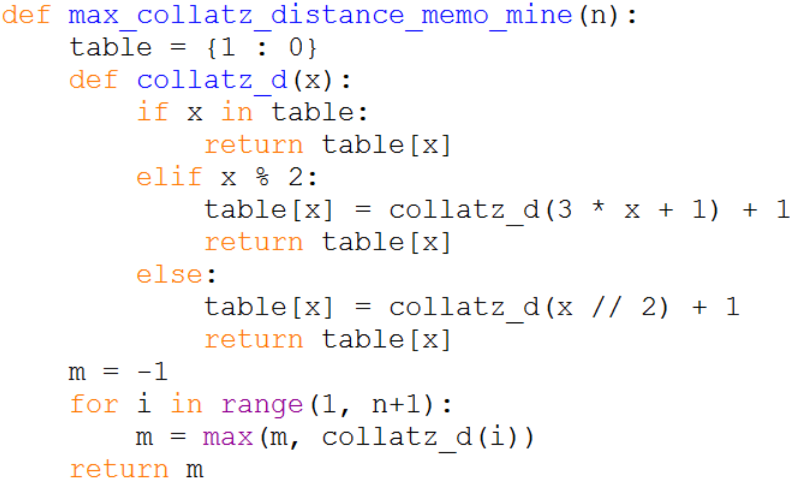
Write your own memoization for *max_collatz_distance*(*n*).

Question 1 Part D Discussion

We can simply follow the recipe:

- Construct a table (dictionary) to record the values
- Construct a new function instead of *collatz_distance*
- Check in the table for the function value before recurse!

Question 1 Part D Zexin's solution



```
def max_collatz_distance_memo_mine(n):
    table = {1 : 0}
    def collatz_d(x):
        if x in table:
            return table[x]
        elif x % 2:
            table[x] = collatz_d(3 * x + 1) + 1
            return table[x]
        else:
            table[x] = collatz_d(x // 2) + 1
            return table[x]
    m = -1
    for i in range(1, n+1):
        m = max(m, collatz_d(i))
    return m
```

Question 2 Part A

Your ability to access a URL on the internet is not guaranteed - it is only on a ‘best effort’ basis. Describe (no need for exact exceptions/error codes) some of the things that could go wrong.

- Error 404, website not found.
- NUS WiFi crashed.
- host server crashed.

Question 2 Part B

Why is it a good idea to raise an error instead of simply returning a string 'Not Found' or an empty string to indicate that the URL is not accessible?

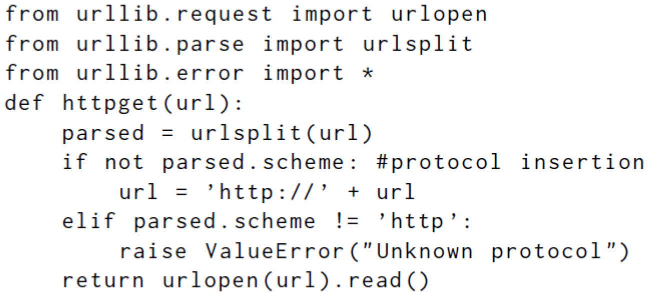
- It is possible that an empty string is an actual response from a server.
- We want better indication of an error/exception.
- Some errors may happen when you are attempting to check for them!

Question 2 Part C

Modify httpget to accomodate the error handling.

- For user errors, we try to catch URLError and rethrow it as ValueError
- For internet error, we try to catch HTTPError and rethrow it as custom error type InternetFail.
- For other errors, just rethrow.

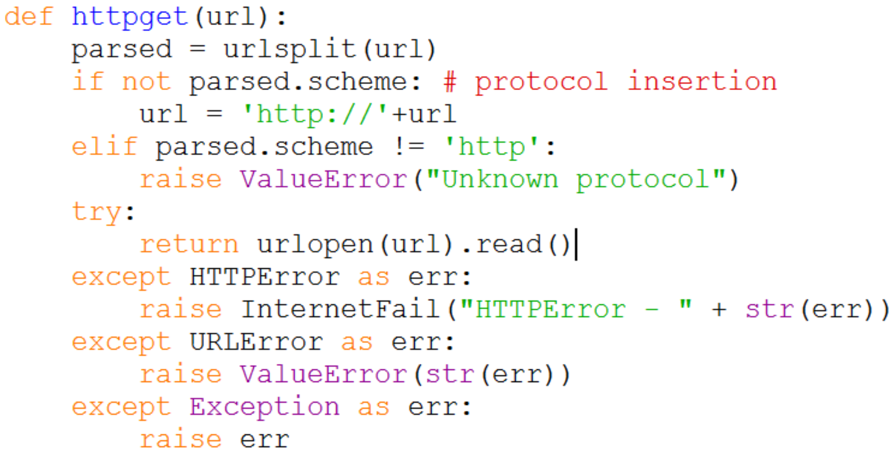
Question 2 Part C Discussion



```
from urllib.request import urlopen
from urllib.parse import urlsplit
from urllib.error import *
def httpget(url):
    parsed = urlsplit(url)
    if not parsed.scheme: #protocol insertion
        url = ’http://’ + url
    elif parsed.scheme != ’http’:
        raise ValueError(”Unknown protocol”)
    return urlopen(url).read()
```

Where will an error/exception possibly happen?

Question 2 Part C Zexin's solution



```python
def httpget(url):
    parsed = urlsplit(url)
    if not parsed.scheme: # protocol insertion
        url = 'http://'+url
    elif parsed.scheme != 'http':
        raise ValueError("Unknown protocol")
    try:
        return urlopen(url).read()
    except HTTPError as err:
        raise InternetFail("HTTPError - " + str(err))
    except URLError as err:
        raise ValueError(str(err))
    except Exception as err:
        raise err
```

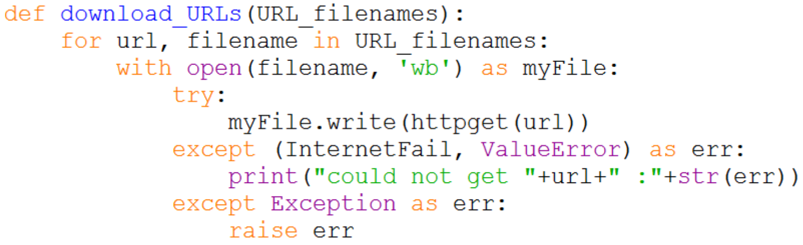
Question 2 Part D

Using the above, write a function download_URLs(URL_filenames) to download a set of files, where URL_filenames is a list of pairs in the following format: [[URL1, filename1], [URL2, filename2], . . .]. In this instance, the contents of URL1 should be saved locally as filename1.

Naturally, many errors can occur during downloading - if we get InternetFail or ValueError, we want to ignore those and continue downloading the rest of the list. Otherwise, we rethrow the error.

Remember how we call this ignoring the error and continue downloading?

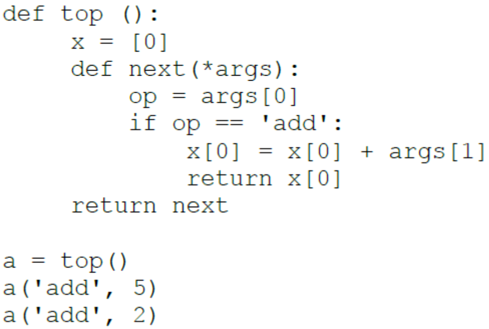
Question 2 Part D Zexin's solution



```python
def download_URLs(URL_filenames):
    for url, filename in URL_filenames:
        with open(filename, 'wb') as myFile:
            try:
                myFile.write(httpget(url))
            except (InternetFail, ValueError) as err:
                print("could not get "+url+" :"+str(err))
            except Exception as err:
                raise err
```

Extra stuff: A simple final question on Message Parsing

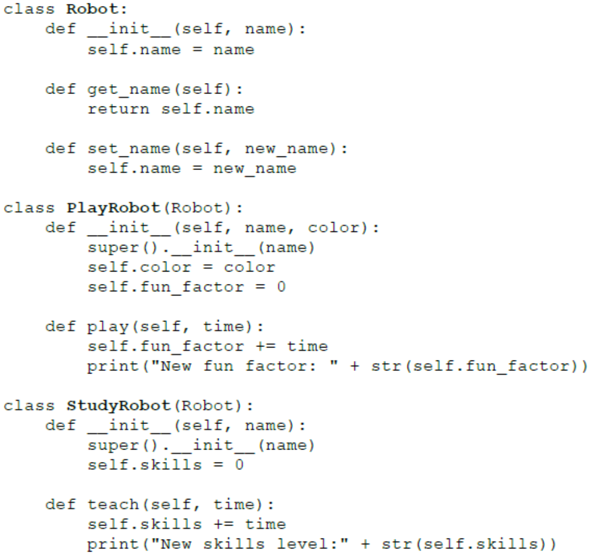
If time permits, we will go through this.



```
def top ():
    x = [0]
    def next(*args):
        op = args[0]
        if op == 'add':
            x[0] = x[0] + args[1]
            return x[0]
    return next

a = top()
a('add', 5)
a('add', 2)
```

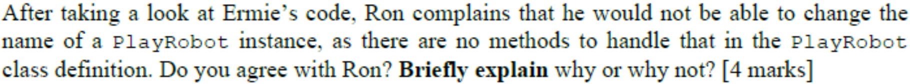
- Remember the recipe for message parsing?
- The answer should be 7.

Extra stuff: A simple final question on OOP



```
class Robot:
    def __init__(self, name):
        self.name = name

    def get_name(self):
        return self.name

    def set_name(self, new_name):
        self.name = new_name

class PlayRobot(Robot):
    def __init__(self, name, color):
        super().__init__(name)
        self.color = color
        self.fun_factor = 0

    def play(self, time):
        self.fun_factor += time
        print("New fun factor: " + str(self.fun_factor))

class StudyRobot(Robot):
    def __init__(self, name):
        super().__init__(name)
        self.skills = 0

    def teach(self, time):
        self.skills += time
        print("New skills level:" + str(self.skills))
```

Extra stuff: A simple final question on OOP

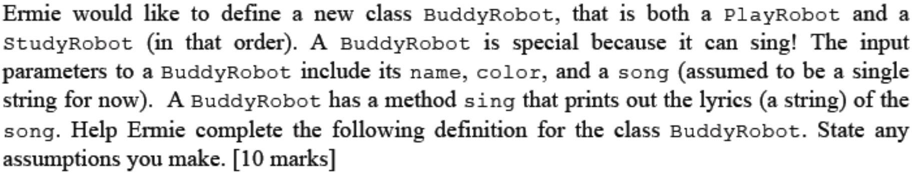


After taking a look at Ermie's code, Ron complains that he would not be able to change the name of a `PlayRobot` instance, as there are no methods to handle that in the `PlayRobot` class definition. Do you agree with Ron? **Briefly explain** why or why not? [4 marks]

Recall inheritance!

This is giving mark.

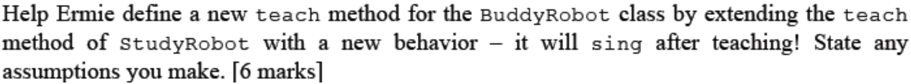
Extra stuff: A simple final question on OOP



Ermie would like to define a new class `BuddyRobot`, that is both a `PlayRobot` and a `StudyRobot` (in that order). A `BuddyRobot` is special because it can sing! The input parameters to a `BuddyRobot` include its `name`, `color`, and a `song` (assumed to be a single string for now). A `BuddyRobot` has a method `sing` that prints out the lyrics (a string) of the `song`. Help Ermie complete the following definition for the class `BuddyRobot`. State any assumptions you make. [10 marks]

Recall multiple inheritance!

Extra stuff: A simple final question on OOP



Help Ermie define a new `teach` method for the `BuddyRobot` class by extending the `teach` method of `StudyRobot` with a new behavior – it will `sing` after teaching! State any assumptions you make. [6 marks]

Recall overriding and calling superclass methods.

Feedback & more

- Slides + relevant material available at:

  `https://github.com/wangzexin/Teaching`

- After the tutorial, if you have further questions:

  `wang.zexin@u.nus.edu`

Thank You

wang.zexin@u.nus.edu