COSC2429 Introduction to Programming

Advanced topics & revision

Outline

- What is recursion?
- Example: Calculating sum of a list of numbers
- Example: Reverse of a string
- Object-oriented programming
- User-defined classes
- Other IT courses that you can take
- Final exam

What is recursion?

- **Recursion** is a method of solving problems that involves breaking a problem down into **smaller and smaller sub-problems of the same type** until you get to a small enough sub-problem that it can be solved trivially.
- Recursion allows us to write elegant solutions to problems that may otherwise be very difficult to program.
- A recursive algorithm must have:
  1. **Base case**: the problem becomes extremely simple in this base case thus it can be solved trivially.
  2. **Recursive case**: the problem is broken into one or more sub-problems of the same type toward the base case.
  3. Each sub-problem can be solved by calling the **recursive function itself**.

Example: Calculate sum of a list of numbers

- We used to calculate the sum of a list of numbers using a loop as follow:

```
def sum(num_list):
    '''
    Calculate the sum of all numbers in the given list
    :param num_list: a list of numbers
    :return: sum of all numbers in num_list
    '''
    total = 0
    for num in num_list:
        total = total + num
    return total

print(sum([1,3,5,7,9]))
```

- Is it possible to write the above function without using a for or while loop?
- Yes, here is how we can think:

  total = 1 + 3 + 5 + 7 + 9 = 1 + (3 + 5 + 7 + 9) = 1 + (3 + (5 + 7 + 9))

  = 1 + (3 + (5 + (7 + 9)))

- Because we know how to add 2 numbers, we can gradually add 7 and 9 first, then add 5 to the result and so on until there is no more items in the list.

Example: Calculate sum of a list of numbers

- Thus we can develop a recursive solution as follow:
  1. Base case: sum(list of one number) = that number
  2. Recursive case: sum(num_list) = first(num_list) + sum(rest(num_list))
- Thus we can re-write the previous function using recursion as:

```
def sum(num_list):
    '''
    Calculate the sum of all numbers in the given list
    :param num_list: a list of numbers
    :return: sum of all numbers in num_list
    '''
    if len(num_list) == 1:
        return num_list[0]
    else:
        return num_list[0] + sum(num_list[1:])

# Test the function
print(sum([1,3,5,7,9]))
```

Example: Reverse of a string

- Write a function that takes a string as a parameter and returns a new string that is the reverse of the old string.
- Let’s call the function as **reverse(s)** in which s is a string parameter.
    1. Base case: reverse("") is the empty string itself
    2. Recursion case: reverse(s) = reverse(s[1:]) + s[0]

```
def reverse(s):
    '''
    Find the reverse of string s
    :param s: a string
    :return: the reverse of s
    '''
    if s == "":
        return s
    else:
        return reverse(s[1:]) + s[0]

# Test the function
print(reverse("abc"))
```

Object-oriented programming

- So far the programs we have been writing use a **procedural programming paradigm** in which the focus is on writing functions (or procedures) which operate on data.
- In **object-oriented programming** the focus is on the creation of **objects** which contain both **data** and the **functions** which operate on those data together.
- Usually, each object definition corresponds to some object or concept in the real world and the functions that operate on that object correspond to the ways real-world objects interact. This is how human think about the world.
- Although Python allows procedural programming, it is actually an object-oriented programming language.
- In Python, every value is actually an object. For example, an integer, a float, a string, a list, etc.
- To create an object, we need an object definition or **class**. Python defined a number of classes for us to use such as **int**, **float**, **str**, **Turtle**, **Screen**, etc.

User-defined classes

- But we can also define our own class as in the following example:

```
class Point:
    """ Point class for representing and manipulating x, y coordinates. """

    def __init__(self):    # constructor is a special method that initialize an object
        self.x = 0
        self.y = 0

# Main program
p = Point()        # Instantiate an object of type Point
q = Point()        # Another object of type Point

print(p)
print(q)

print(p is q)
```

User-defined classes

- Let’s extend our class by adding more methods and parameters:

```
class Point:
    """ Point class for representing and manipulating x, y coordinates. """

    def __init__(self, init_x, init_y):
        self.x = init_x
        self.y = init_y

    def get_x(self):
        return self.x

    def get_y(self):
        return self.y

    def distance_from_origin(self):
        return ((self.x ** 2) + (self.y ** 2)) ** 0.5

# Main program
p = Point(7, 6)
print(p.get_x(), p.get_y())
print(p.distance_from_origin())
```

User-defined classes

```
class Point:
    """ Point class for representing and manipulating x, y coordinates. """

    def __init__(self, init_x, init_y):
        self.x = init_x
        self.y = init_y

    def get_x(self):
        return self.x

    def get_y(self):
        return self.y

    def distance_from_origin(self):
        return ((self.x ** 2) + (self.y ** 2)) ** 0.5

    def distance(point1, point2):
        x_diff = point2.get_x() - point1.get_x()
        y_diff = point2.get_y() - point1.get_y()
        return (x_diff**2 + y_diff**2) ** 0.5

# Main program
p = Point(4, 3)
q = Point(0, 0)
print(distance(p, q))
```

Other IT courses that you can take

IT course you can take after COSC2429:

- COSC2430 Web Programming (HTML, CSS, JavaScripts, Python)
- COSC2081 Programming 1 (Java)
- ISYS2089 Software Engineering Fundamentals
- ISYS2077 Database Concepts

Popular IT courses as General Electives:

- COSC2634 Building IT Systems
- COSC2652 User-Centred Design
- COSC2500 Introduction to Computer System and Platforms

Final exam

- Time: 14.00 – 16.15 Friday of week 13 (May 20, 2016)
- Location: 2.4.02 and 2.4.03
- You will be allowed to enter the exam room around 13:45. Don’t be late!!!
- 50% of the final result
- 50 points, 2 problems, 25 points/problem
- 15 minutes reading + 120 minutes writing on a lab computer
- Cover topics from week 1 to 10

Topics

1. Variable and basic data types
2. Turtle graphics module
3. Other Python modules
4. Functions
5. Iterations
6. Selections
7. Strings
8. Lists
9. Files
10. Dictionaries

General exam instructions

1. You must bring student ID to the exam. Without a student ID, you will not be allowed to enter the exam room.
2. Books, mobile phones, calculators, USB flash drives and other electronic devices are not allowed in the exam room. However, **lecture slides** and **dictionaries** are allowed in the exam room.
3. Writing is not allowed during reading time. During reading time, students may ask an invigilator to clarify questions.
4. At the start of the writing time, write your student ID, name and signature in the space provided below.
5. When writing code, observe the usual style guidelines for meaningful names and layout.
6. Clearly state any assumptions you make in reaching your answers.
7. Answer all questions. Plan your time accordingly.
8. Students must be silent **at all times** in the exam room and remain seated while the invigilators are collecting the exam papers and draft papers.
9. The general Rules of Conduct apply.

Exam submission instructions

1. At the start of the writing time, you must login a lab computer with **a designated exam account** specified by the invigilator. If you log in with a different account, you will be charged with plagiarism.
2. At the end of the writing time, you must submit **two Python files** by login Blackboard with your student ID and password then uploading them to the **Final Exam** in the **Assessments** section of this course. Make sure to check carefully that the Python files submitted are the correct ones. If Blackboard is not available at that time for any reasons, the invigilator will copy your exam files to his/her USB flash drive.
3. After submitting your files successfully, you must delete them in the computer and then log out. The exam paper and any draft papers must be handed in.

How to prepare for the final exam?

- Review all lecture slides
- Read the interactive online textbook
- Redo all the tutorial exercises
- Review your assignments
- If you need help:
  - Post any questions that you may have in the Discussion Board
  - Meet me in person during my consultation hours in week 13