

Tutorial 5: Duck Typing and PA2

CSCI3180 Teaching Group

My Information

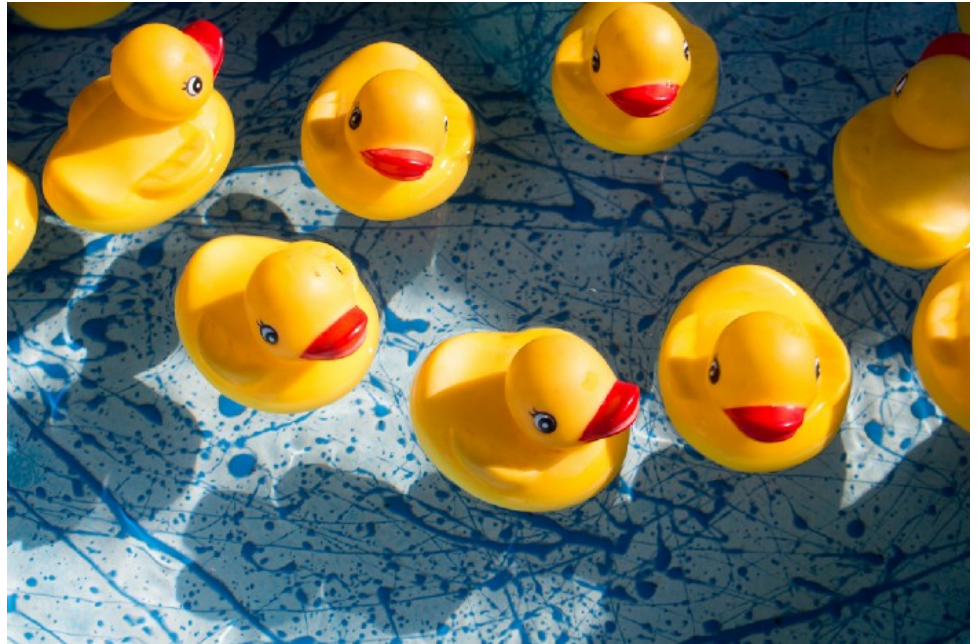
- Juzheng Miao
- Office:
 - SHB 1024 A
- Office Hours:
 - Tuesday 10.30AM - 11.30AM
- jzmiao22@cse.cuhk.edu.hk

Outline

- Duck Typing
- Assignment 2 preview

Duck Typing

Duck Typing is a term commonly related to **dynamically-typed** programming languages and **polymorphism**. The idea behind this principle is that the code itself does not care about whether an object is a **duck**, but instead it does only care about whether it **quacks**.



If it walks like a duck, and it quacks like a duck, then it must be a duck.

Understanding Duck Typing

- Three classes: Duck, Goose, and Fish
- Custom types are more about **implementing features** than data types. Even though a goose isn't an actual duck, the swim_fly function turns it into one.

```
class Duck:
    def swim_fly(self):
        print('I am a duck, and I can swim and fly.')
class Goose:
    def swim_fly(self):
        print('I am a goose, and I can swim and fly.')
class Fish:
    def walk(self):
        print("I am a fish, and I can swim but can't fly.")

for obj in Duck(), Goose(), Fish():
    obj.swim_fly()
```

```
I am a duck, and I can swim and fly.
```

```
I am a goose, and I can swim and fly.
```

```
Traceback (most recent call last):
```

```
  File "/Users/yuanyuchen/Documents/my/pythonproject/test1.py",
    obj.swim_fly()
```

```
AttributeError: 'Fish' object has no attribute 'swim_fly'
```

Practical Examples---Iteration

- For iteration the class must have `__iter__()` and `__next__()` functions which makes it eligible for iterating.
- We can even define our **own iterator** for printing square numbers, where we define methods `__iter__()` and `__next__()` which are two methods called during python iteration.

```
class Squares:
    def __init__(self, l=0, u=-1):
        self.u = u
        self.n = l

    def __iter__(self):
        return self

    def __next__(self):
        if self.n < self.u:
            s = self.n ** 2
            self.n += 1
            return s
        else:
            raise StopIteration
```

```
for i in Squares(1, 4):
    print(i)
```

```
1
4
9
```

Practical Examples---len()

- The len() function can be used with any object implementing the `__len__` method.
- The object's type itself is not significant. Therefore, we do not declare the argument in method prototypes.
- What really matters is if the object has particular attributes (i.e., `__len__` method) at run time.

```
class Specialstring:
    def __len__(self):
        return 21

string = Specialstring()
print(len(string)) # 21
```

Practical Examples---callable()

- Anything with a `__call__` is a callable.
- Invent your own callable objects.

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

    def __call__(self):
        print("My name is", self.name)

>>> trey = Person("Trey")
>>> trey
<__main__.Person object at 0x7f9ddd78e0b8>
```

- The Person object is also callable! We can call that Person object by putting parentheses after it:

```
>>> trey()
My name is Trey
```


Assignment 2 preview

Introduction

In this assignment, you are going to write a python program to implement the rules of a single-player game named *Greedy Snake* based on the given C++ implementation. Moreover, we will extend the game and you need to implement more classes for the game. *Note: All the Python implementations in this assignment should be built using any [Python version after \(including\) 3.6](#).*

The main objectives of this assignment are three-fold for you:

- To practice Python and learn the differences between Python and C++.
- To explore the pros and cons of dynamic typing in Python.
- To better understand duck typing in Python and know how to properly use it.

Requirement: Python \geq 3.6

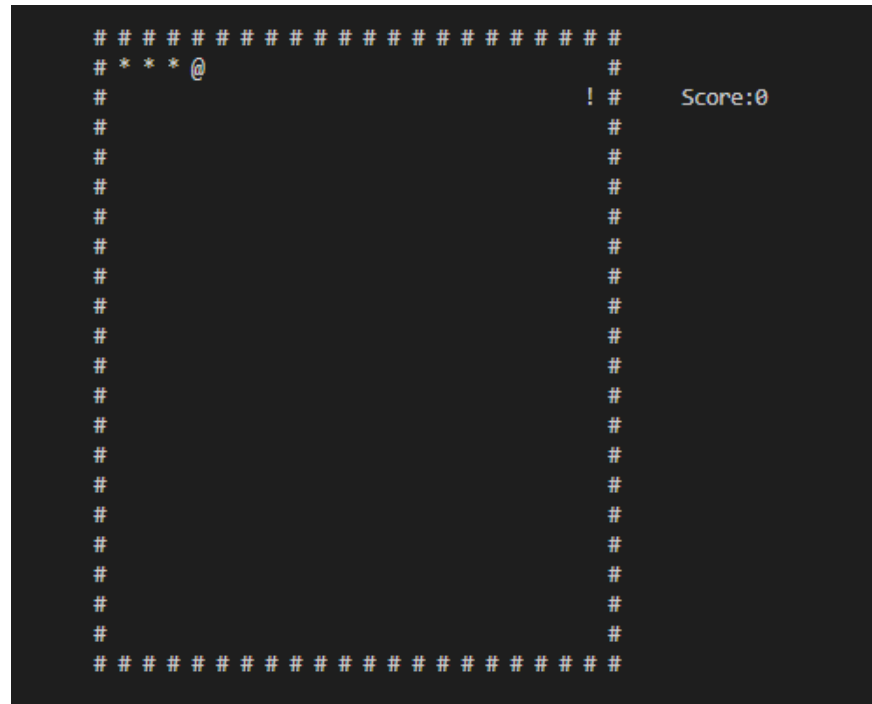
Greedy Snake

- Greedy Snake is a single-player game.
- Try to eat more and grow up to the maximum length.
- Don't crash into the wall and yourself.



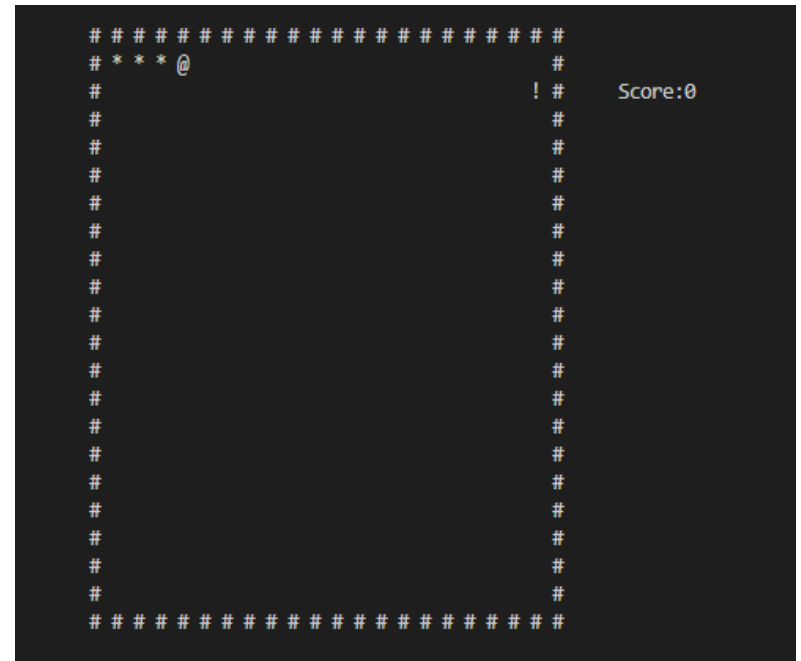
Game Board

- The board is a 22 x 22 grid.
- The snake appears at the top-left corner of the board, with an initial length of 3.
 - snake head: '@'
 - snake body: '*'



Food

- Base_score: 100
- Apple and Water
 - Apple ('o'): _maturity (boolean)
 - Water ('!'): _volume (integer)
- Actual_score
 - Apple:
actual_score=base_score
if *_maturity* else actual_score= 0
 - Water:
actual_score=base_score * _volume



Move

- By inputting 'w', 'a', 's', and 'd' to move a snake around the board.
 - 'w': up, 's': down, 'a': left, 'd': right.
- The default moving direction is right, if an invalid key (a key not in the 4 pre-defined keys) is pressed.

[illegible]

Game Over

- The snake crashes into the wall or itself. Lose ☹️
- The snake grows up to the maximum length. Win 😊

```
# # # # # # # # # # # # # # # #  
#                                     #  
#                               ! #   Score:0  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#                                     #  
#          *                            #  
#         *                            #  
#         *                            #  
##### @ #####
```

Game over!
Your final score is: 0
Sad! You die before growing up.

```

# # # # # # # # # # # # # # # # # #
#                                     #
#                                     #
#                                     * * !
#                                     *
#                                     *
#                                     *
#                                     *
#                                     *
#                                     *
#                                     *
#                                     *
#                                     *
#                                     *
#                                     *
#                                     *
#                                     *
#                                     @
#                                     #
#                                     #
#                                     #
#                                     #
#                                     #
# # # # # # # # # # # # # # # # # #
Game over!
Your final score is: 1500
Congratulations! You successfully grow up.

```

```

#####
#
#
#      * @ *
#      * *
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#####
Score:100

```

ore

Game over!

Your final score is: 100

Sad! You die before growing up.

Task 1&2: OOP & Dynamic Typing

- Task 1 requires you to implement the game using Python.
 - The C++ implementation and Python skeleton is given to you for reference.

Sample output & grading

Your finished program should produce the same output as the following screenshots for each step. But you don't need to worry about the printing format, we have provided the printing functions. What you need to do is to implement some important functions and make the generated information correct in each step.

First, in each step, your snake should be able to move using the pre-defined 4 keys ('w', 'a', 's', and 'd'). Related information are also correctly printed.

Please select the moving direction! (w : up, s : down, a : left, d : right)

5

[illegible]

Score:0

Second, after the food is eaten, your snake should be longer. And the score should be also correctly updated. At the same time, a new food is randomly generated at a new place on the board.

Please select the moving direction! (w : up, s : down, a: left, d: right)

5

[illegible]

#

#

#

#

#

#

#

#

#

#

#

#

#

#

#

#

#

#

#

#

#####

#



#



Score: 500



Third, you lose the game and the game is over when the snake either crashes into a wall or moves into itself. Your game should stop and the related information should be correctly printed.

[illegible][illegible]

Fourth, when your snake grows up to the maximum length, your game should stop and the related information should be correctly printed.

```

#####
#
#
#           * *           !           #
#           *           #
#           *           #
#           *           #
#           *           #
#           *           #
#           *           #
#           *           #
#           *           #
#           *           #
#           *           #
#           *           #
#           *           #
#           *           #
#           @           #
#           #           #
#           #           #
#           #           #
#           #           #
#####
#####

```

Game over!

Your final score is: 1500

Congratulations! You successfully grow up.

Task 1&2: OOP & Dynamic Typing

- Task 2 requires you to analyze dynamic typing used in your Python implementation.
 - write a report to demonstrate the advantages and disadvantages of Dynamic Typing.

Dynamic typing is an important characteristic of Python. It has many commonly-claimed advantages:

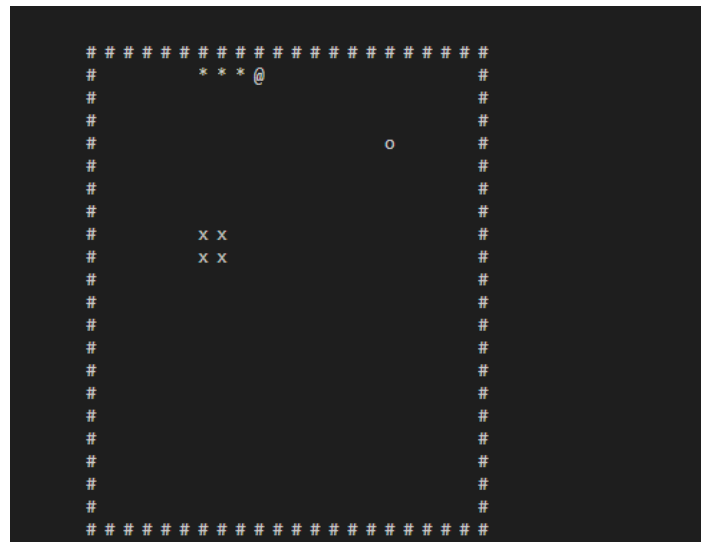
1. Don't need to declare variables or their data type before assignment.
2. It makes functions more reusable because functions can be applied on arguments of different types.
3. ...

However, type checking can only be carried out at runtime, incurring time overhead and reliability issues.

Please provide at least one concise example code with explanation in the context of this assignment to demonstrate the advantages of Dynamic Typing and at least one concise example code with explanation in the context of this assignment to demonstrate the disadvantages of Dynamic Typing.

Greedy Snake with Extension

- The snake catcher shows up!
- Can appear randomly on the board, and has an active area (a 2x2 square region).
- Once the snake crashes into the snake catcher, the total score obtained by the player is deducted by 100.
- The snake catcher's position changes randomly when the snake came into it or after a fixed time interval (20 steps).



Task 3&4: Duck Typing

- Task 3 requires you to extend the game with *SnakeCatcher* class.
 - There are at least two places where duck typing is applicable.
- Task 4 requires you to write a report to compare your Python implementation and our C++ implementation and explain how duck typing makes coding more flexible and convenient.
- For grading, we will consider your program output and the **code implementation details**. Make sure your program can run successfully without compilation errors and crashes by using any **Python version after (including) 3.6**.

```
Please select the moving direction! (w: up, s : down, a: left, d: right)
d

#####
#          * * * @           #
#                                           #
#                                           #
#                                           #
#                   o           #
#                                           #
#                                           #
#                                           #
#               x x             #
#               x x             #
#                                           #
#                                           #
#                                           #
#                                           #
#                                           #
#                                           #
#                                           #
#                                           #
#                                           #
#                                           #
#####
Name: Apple      Position: (4, 16)
Maturity: True   Score: 100

Here is the snake catcher!      Position: (8, 6)

Current Score: 0      Current Length: 4
```

- Your code should be able to generate the snake catcher correctly according to rules we have mentioned above when the snake comes into it or after a fixed time interval (20 steps).
- When the snake crashes into the snake catcher, the score of the snake should be reduced by 100 correctly.
- The display information of each class should be correctly printed as we do in the image. (Don't need to worry about the printing format)


Written report

Write a report to comment on the pros and cons of dynamic typing and duck typing in one PDF file named exactly **a2.pdf**, with at most four A4 pages, containing two main sections:

1. Provide example code and necessary elaborations in the context of this assignment for demonstrating advantages and disadvantages of Dynamic Typing as specified in Task 2.
2. Provide example code and necessary elaborations in the context of this assignment for demonstrating advantages of Duck Typing as specified in Task 4.

Submission & deadline

Submission deadline: 23:59:00, Mar 14 (Tuesday).

- Please submit two python files and one PDF file only: **greedySnake.py**, **greedySnake_enhanced.py**, and **a2.pdf** ONLY.
- Make sure you have completed the *declaration* at the top of the **greedySnake.py** file.
- Make sure your two submitted files can be compiled and run with no errors under our official grading environment (see the [Sample output & grading](#) section).
- Do NOT zip the files. Submit them individually in the same submission. Follow these steps:
 1. Go to the CSCI3180 page on [Blackboard](#).
 2. Go to "Assignment Submission" on the left-hand side menu.
 3. Pick "Assignment 2."
 4. For each of the files that you need to submit, click "Browse Local Files", and then browse for it.
 5. The files will be added to the "Attached files" list. Please **check carefully their filenames**. They must be exactly the same as the ones specified. Wrongly named files won't be graded as most grading are automatic.
 6. Click the blue "Submit" button at the bottom of the page.
 7. Now you **must verify** all of your submitted files by downloading them back and make sure they are your own and the latest version. You can download each of the files with the download button . Click "Start New" to submit again if you need to modify your submission.

- You can submit as many times as you want, but **only the latest submission attempt will be graded**. You do NOT get to choose which version we grade.
- If you submit after the deadline, late penalty will be applied according to the submission time of your last submission attempt.
- **Submit early to avoid any last-minute problem.** Only Blackboard submissions will be accepted. Again, you can submit as many times as you want, so you may make an early submission first well before the deadline, and update it later.
- Late submission penalty is 1 point per 5-minutes late. Less than 5-minutes late is considered as a full 5-minutes late. However, your score will be lower-bound by 0 after any mark deduction.

END

- Q&A