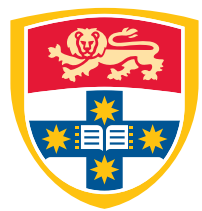
INFO1110 & COMP9001: Introduction to Programming

School of Information Technologies, University of Sydney



Copyright Warning

Lecture 22: More classes and methods

Creating classes, methods and testing

Define a WorldPoint class

Let WorldPoint represent a geographical coordinate on the surface of the earth[1]

Stores a name AND two floating point numbers to represent the coordinate

Default values are Greenwich (51.48, 0)

Can *optionally* be initialised with values

Values are read/write, but only through supported operation (methods)

WorldPoint has range restrictions

- latitude - South to North is [-90, 90]
- longitude - West to East is [-180, 180][2]

---

[1] biaxial ellipsoid

[2] be careful

WorldPoint operations

Report the Latitude

Report the Longitude

Report the name

Report the name, Latitude and Longitude as a formatted String

Report both Latitude and Longitude as a list

Set Latitude

Set Longitude

Report the Euclidean distance of this WorldPoint to another WorldPoint

Using WorldPoint

Write a program to

- Construct 3 WorldPoint objects
- Initialise each WorldPoint using command line arguments
- Print the information of the WorldPoint

```
~> python worldpoint.py Sydney 33.87S 151.21E Moscow 55.75N 37.62E
Manitoba 49.54N 97.08W
Sydney::Latitude: -33.87 Longitude:151.21
Moscow::Latitude: 55.75 Longitude:37.62
Manitoba::Latitude: 49.54 Longitude:-97.08
```

Reference value

When calling a function, we *copy* the value to be used in the function

```
1  def print_plus_one(x):
2      x += 1
3      print(x)
4
5  num = 75
6  print_plus_one(num)
7  print(num)
```

When creating an Object we have a variable that stores a reference, the value of memory address

```
1  wp = WorldPoint("Beijing", 39.92, 116.38)
```

Reference value

When calling a function, the reference value is copied.

```
1  def init_data(data, offset):
2      i = 0
3      while i < len(data):
4          data[i] = i + offset
5          i = i  + 1
6          offset = offset + 1
7
8  offset = 10
9  numbers =  [0] * 10
10 init_data(numbers, offset)
11 print("offset: {} 6th number: {}".format(offset, numbers[5]))
```

Good news: functions that have a reference can make changes to the object

Bad news: functions that have a reference can make changes to the object

Reference value (cont.)

More bad news

We need to also know if the arguments to the function are modified in the first place. That is, if the data type is mutable, it can be modified. Otherwise we cannot know the outcome of a function call on that object.

```
1 s = "hello {}"
2 s.format(123)
```

returns a new string, it does not modify the existing string.

```
1 x = 1
2 x.__add__(1)
```

returns a new int, it does not modify the existing int

Reference value

When calling a method, the reference value is copied.

```
1  def init_data(data, offset):
2      n = len(data)
3      data = [0] * 10
4      i = 0
5      while i < n:
6          data[i] += offset
7          i = i  + 1
8
9  numbers = [0] * 5
10 offset = 7
11 init_data(numbers, offset)
12 print("offset: {} 3rd number: {}".format(offset, numbers[2]))
```

What is the output here?

Understanding references and function calls

When calling a method, the reference value is copied

```
1  def foo(a, b):
2      # cannot see inside magic function
3
4  (x, y) = get_parameters()
5  foo(x, y)
```

Were operations with object's `a` or `b` modified during execution of `foo()`?

Were object's `x` or `y` modified after calling `foo()`?

Understanding references and function calls (cont.)

We want to know the state of the program at any given time

We use the idea of a desk check to understand the changes from one instruction to the next

Problem: we don't know if after calling foo(a,b) what the state is.

To confirm what is the state of the program after a function call that uses objects we need more information:

1. We need to know if the data type mutable or not
2. If it is mutable, we need to know if function `foo()` calls methods on mutable object
3. If it is mutable, and that object has a method called upon it, we need if that method modifies the state of the object

Storing multiple return values using reference

Previously seen, values can be stored in a reference type

```
1  def get_quad_roots(a, b, c, roots):
2  '''returns roots of quadratic equation ax^2 + bx + c = 0
3     roots has at least two elements. Throws TypeError exception
          when roots is not a list, ZeroDivisionError if a is zero,
          ValueError when discriminant less than zero, and
          IndexError if list length is less than 2'''
4
5  p = [1, 5, 1]
6  roots = [0,0]
7  get_quad_roots(p[0], p[1], p[2], roots)
```

Storing multiple return values using reference (cont.)

The same can apply to any Object

```
1 def worldpoint_set_location(point, hLatitude, hLongitude):
2 '''converts a human version of latitude/longitude to numeric form
      and sets those values in the WorldPoint object. throws
     ValueError on failed number conversion or IndexError on
     exception if incorrect numbers'''
```

The above is NOT a method. It was not a supported operation of WorldPoint object. Does it belong in the class WorldPoint?

class variables

Variables that belongs to the `class`.

Instance variables belong to *one* instance, whereas class variables are common to *all* instances.

What is common to all objects? Identifiers, global values, shared settings

```
1  class Student:
2      # global counter for ID
3      sid = 0
4
5      def __init__(self, name):
6          self.name = name
7          self.id = Student.sid
8          Student.sid += 1
9
10 students = [Student("Kerry"), Student("Xixi"), Student("Alice")]
11 for student in students:
12     print(student.id)
```

static and class methods

static or class methods can always be called without any objects ever being created.

instance methods with `self` must be associated with the memory of an instance.

static or class methods can be used to operate on class variables, or they can perform operations related to the class similar to a function (input, process, output)

```
1  class WorldPoint:
2      ...
3
4      # Latitude/Longitude converted to read as xxx S/N  yyy W/E
5      # returns a list of 2 strings. Each is positive num
6      # 1st element has S/N, 2nd element W/E
7      def get_human_readable(worldpoint):
8          ...
9      def get_human_readable(lattitude,longitude):
10         ...
11 }
```

static and class methods (cont.)

What about students?

```
1  class Student:
2      # global counter for ID
3      sid = 0
4
5      def __init__(self, name):
6          self.name = name
7          self.id = Student.sid
8          Student.sid += 1
9
10     def get_students_created():
11         return Student.sid
12
13     def enrolled_first(student_a, student_b):
14         '''returns True if student a enrolled before student b.
               False otherwise. Exceptions are thrown in error cases
               '''
15         return student_a.id < student_b.id
16
17 students = [Student("Kerry"), Student("Xixi"), Student("Alice")]
18
```

static and class methods (cont.)

```
19 answer = Student.get_students_created()
20 print("Number of students ever created: {}".format(answer))
21
22 xixi = students[1]
23 is_xixi_first = False
24 for student in students:
25     if student.name == "Xixi":
26         continue
27     if not Student.enrolled_first(xixi, student):
28         is_xixi_first = False
29
30 if is_xixi_first:
31     answer = ""
32 else:
33     answer = "not "
34 print("Xixi is {}first".format(answer) )
```

Review

class the type of an object, e.g., the class `str`; to do with the type of an object, e.g. class variable or class method;

`object` the most basic class in `Python`;

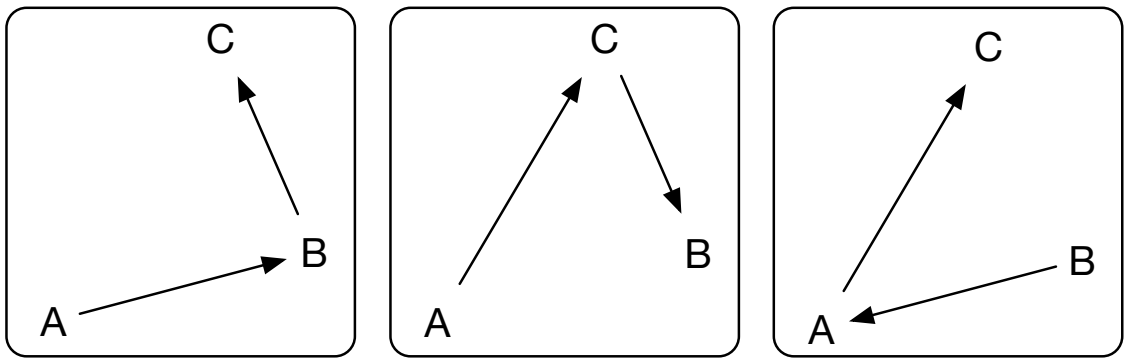
instance to do with a single copy or case of, e.g. "piano" is an *instance* of the type "MusicInstrument"

method a separate block of code that can be called, e.g., `s.format(...)` or `s.__len__()` for a string `s`;

class/static variable/method in the current context, applying to the whole class, as variables, e.g. `sys.argv`), or as methods, e.g., `str.split("simple word", " ");`

WorldPoint Shortest path

Calculate the shortest distance path of the three WorldPoint's and when found, print the name of the point and the distance from the previous point.




```
~> python worldpoint.py Sydney 33.52S 151.13E Perth 31.95S 115.86E
Brisbane 27.47S 153.03E
Perth 0
Sydney 17542.643054323693
Brisbane 23145.74687879746
```

Can Euclidean distance measure with these values lead to false result?