CSIT110

Fundamental Programming with Python

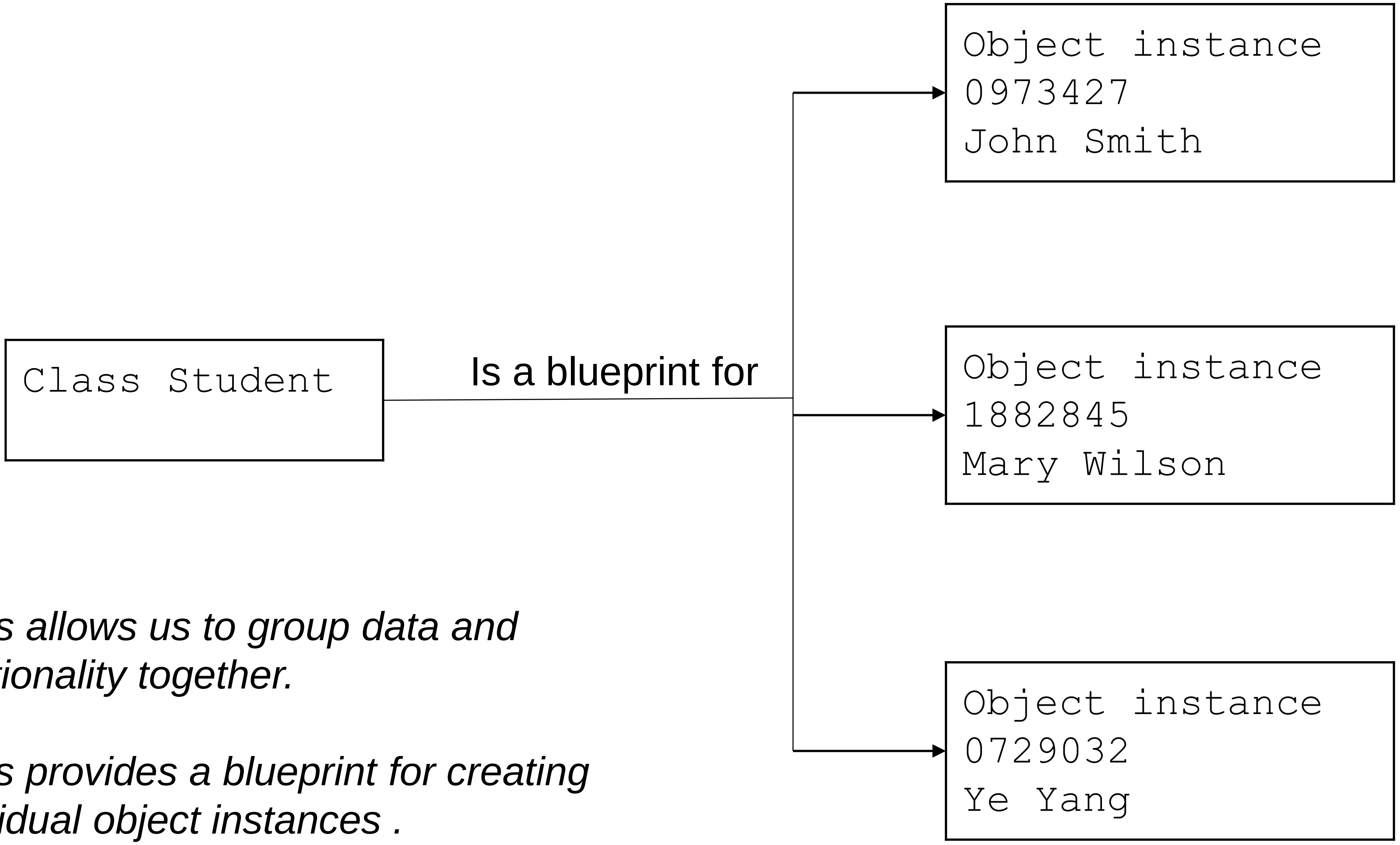
Class and Object

Goh X. Y.

In this lecture

- Class and Object
    - Instance attribute
    - Class attribute
    - Instance method
    - Special (dunder) method
- Class inheritance

Class and Object



Class allows us to group data and functionality together.

Class provides a blueprint for creating individual object instances .

How does a Class look like

```
class ClassName:                                              Use Camelcase
  """ Documentation
      description of the Class
  """
  class_variable1 = "Fixed text"  # class variable shared by all instances
  class_variable2 = 6248
  class_variable3 = "the list goes on"
  def __init__(self, id, first_name, last_name):
    # things to be done such as instantiating instance variables
    # when an object instace of this class is created goes here
    self.first_name = first_name  # instance variable unique to each instance
    ...
  def class_method1(self):
    # do something

  def class_method2(self): # do something
```

Example

In a University called Solla Sollew where each student has a\an

- unique student id
  - e.g. student `John Smith` has student id `0973427`)
- username
  - constructed from the first name initial, last name initial and the first 3 digits of the student id
  - e.g. `John Smith`'s username is `js097`
  - constructed at the enrolment day
  - will **never be changed** even though student may change their name.
- Unix home directory
  - e.g. `John Smith` home directory is `/user/student/js097`)
- an email
  - it will never be changed even though student may change their name
  - e.g. `John Smith`'s email is `js097@solla.sollew.edu`
- an email alias
  - e.g. `John Smith`'s email alias is `John.Smith.097@solla.sollew.edu`).
  - Changes automatically when student changes name.
  - e.g. for example, if `John Smith` last name changed to `Lee` then his email alias is automatically changed to `John.Lee.097@solla.sollew.edu`)

Instance attribute vs Class attribute

Instance attribute: data belong to individual object instance.

Class attribute: data that is common to all objects. (Some classes do not have any class attributes.)

For example,

- Each student object has its own first name, last name and student id, etc...
    - student id `0973427`, first name `John`, last name `Smith`, username `js097`, ...

- All students share the same email domain and Unix student directory.
    - Email domain `solla.sollew.edu`, Unix student directory `/user/student`, ...

Defining class and creating object

```
class Student:
  """
  The class, Student, represents a student
  with the following attributes:
    id: student number
    first_name: first name
    last_name: last name
    username: Unix account username
  """

  email_domain = "solla.sollew.edu"
  student_dir = "/user/student"

  def __init__(self, id, first_name, last_name):
    ...
```

class attributes

instance attributes

Defining class and creating objects

```
class Student:
  def __init__(self, id, first_name, last_name):
    self.id = id
    self.first_name = first_name
    self.last_name = last_name

    # username is constructed in the beginning
    # and will not change if name changed
    # username = lowercase initials + first 3 id digits
    self.username = first_name[0].lower() + last_name[0].lower() + id[0:3]
```

instance attributes

```
# creating 3 student objects
student1 = Student("0973427", "John", "Smith")

student2 = Student("1882845", "Mary", "Wilson")

student3 = Student("0729032", "Ye", "Yang")
```

Defining class and creating objects

```
# Defining the class - Student
class Student:
  def __init__(self, id, first_name, last_name):
    ...
```

```
# creating 3 student objects
student1 = Student("0973427", "John", "Smith")

student2 = Student("1882845", "Mary", "Wilson")

student3 = Student("0729032", "Ye", "Yang")
```

Instance method, `exampleFcn(self, args) :`

- Automatically pass the object instance (**`self`**) as the first parameter e.g: `"text".upper()` or `"text".find("ex")`

Defining class and creating objects

```
# Defining the class - Student
class Student:
  def __init__(self, id, first_name, last_name):
    ...
```

```
# creating 3 student objects
student1 = Student("0973427", "John", "Smith")

student2 = Student("1882845", "Mary", "Wilson")

student3 = Student("0729032", "Ye", "Yang")

# get object attributes
print(student1.id)
print(student1.first_name)
print(student1.last_name)
print(student1.username)
```

Accessing class attributes

```
class Student:

  email_domain = "solla.sollew.edu"
  student_dir = "/user/student"

```

```
# creating 3 student objects
student1 = Student("0973427", "John", "Smith")
student2 = Student("1882845", "Mary", "Wilson")
student3 = Student("0729032", "Ye", "Yang")

# get class attributes
print(student1.email_domain)        # not recommended
print(student2.email_domain)        # not recommended
print(student3.email_domain)        # not recommended

print(Student.email_domain)         # recommended
```

Modifying object instance attributes

```
student2 = Student("1882845", "Mary", "Wilson")

# display object attributes
print("Before: ")
print(student2.id)
print(student2.first_name)
print(student2.last_name)

# change student last name
student2.last_name = "Davis"

# display object attributes after update
print("After: ")
print(student2.id)
print(student2.first_name)
print(student2.last_name)
```

```
Before:
1882845
Mary
Wilson
```

```
After:
1882845
Mary
Davis
```

Modifying class attributes

```
# change email domain
Student.email_domain = "mail.solla.sollew.edu"

# change student directory
Student.student_dir = "/usr/home/student"
```

Use Class name and not the name of the object instance

Defining an object instance method

```
class Student:
    def fullname(self):
    """
    Get student's full name
    """
    return self.first_name + " " + self.last_name
```

Instance method:

- Automatically pass the object instance (**`self`**) as the first parameter
- May use instance attribute and instance method

Defining an object instance method

```
class Student:
    def fullname(self):
    """
    Get student's full name
    """
    return self.first_name + " " + self.last_name
```

```
# creating a student object
student1 = Student("0973427", "John", "Smith")

# calling method - from the object instance
print(student1.fullname())          # recommended

# calling method - from the class
print(Student.fullname(student1))   # not recommended
```

Defining an object instance method

```
class Student:
  def fullname(self):
    return self.first_name + " " + self.last_name
```

```
# creating a student object
student2 = Student("1882845", "Mary", "Wilson")

# display object attributes
print("Before: ")
print(student2.fullname())

# change student last name
student2.last_name = "Davis"

# display object attributes after update
print("After: ")
print(student2.fullname())
```

```
Before:
Mary Wilson
After:
Mary Davis
```

Defining an object instance method

```
class Student:
  def email(self):
    """
    Get student's email: username@domain
    """
    return self.username + "@" + Student.email_domain
```

```
# creating a student object
student2 = Student("1882845", "Mary", "Wilson")

# display email
print(student2.email())
```

```
mw188@solla.sollew.edu
```

Defining an object instance method

```
class Student:
  def email_alias(self):
    """
    Get student's email: username@domain
    """
    return self.first_name + "." + self.last_name +
           "." + self.id[0:3] + "@" + Student.email_domain
```

```
# creating a student object
student2 = Student("1882845", "Mary", "Wilson")

# display email
print(student2.email_alias())
```

```
Mary.Wilson.188@solla.sollew.edu
```

Defining an object instance method

```
class Student:

  def home_dir(self):
    """
    Get student's Unix home directory:
    studentDir/username
    """
    return Student.student_dir + "/" + self.username
```

```
# creating a student object
student2 = Student("1882845", "Mary", "Wilson")

# display home directory
print(student2.home_dir())
```

```
/user/student/mw188
```

Defining an object instance method

```
class Student:

  def print_detail(self):
    print("Student ID: " + self.id)
    print("First name: " + self.first_name)
    print("Last name: " + self.last_name)
    print("Full name: " + self.fullname())
    print("Username: " + self.username)
    print("Email: " + self.email())
    print("Email alias: " + self.email_alias())
    print("Home directory: " + self.home_dir())
```

```
# creating a student object
student2 = Student("1882845", "Mary", "Wilson")

# display details
print(student2.print_detail())
```

Defining an object instance method

```
# creating a student object
student2 = Student("1882845", "Mary", "Wilson")
print("Before:")
print(student2.print_detail())

# change student last name
student2.last_name = "Davis"
print("After:")
print(student2.print_detail())
```

```
Before:
Student ID: 1882845
First name: Mary
Last name: Wilson
Full name: Mary Wilson
Username: mw188
Email: mw188@solla.sollew.edu
Email alias: Mary.Wilson.188@solla.sollew.edu
Home directory: /user/student/mw188
```

Defining an object instance method

```
# creating a student object
student2 = Student("1882845", "Mary", "Wilson")
print("Before:")
print(student2.print_detail())

# change student last name
student2.last_name = "Davis"
print("After:")
print(student2.print_detail())
```

```
After:
Student ID: 1882845
First name: Mary
Last name: Davis
Full name: Mary Davis
Username: mw188
Email: mw188@solla.sollew.edu
Email alias: Mary.Davis.188@solla.sollew.edu
Home directory: /user/student/mw188
```

Special (dunder) method

Starts with double underscores e.g. `__methodName__`

Special (dunder) method

We have seen a special (dunder) method:

```
class Student:
  def __init__(self, id, first_name, last_name):
```

Now we will write another special (dunder) method:

```
class Student:
  def __str__(self):
```

Why do we need this method **`__str__`** ?
Try this and see the result:

```
student2 = Student("1882845", "Mary", "Wilson")
print("Object student2 is " + str(student2))
```

```
Object student2 is <__main__.Student object at 0x7f282523ecf8>
```

Special (dunder) method

```
class Student:

  def __str__(self):
    return "{0} ({1})".format(self.fullname(), self.id)
```

Now try this and see the result:

```
student2 = Student("1882845", "Mary", "Wilson")

print("Object student2 is " + str(student2))
```

```
Object student2 is Mary Wilson (1882845)
```

Special (dunder) method

```
class Student:
  def __repr__(self):
    return "Student('{0}', '{1}', '{2}')" \
        .format(self.id, self.first_name, self.last_name)
```

```
student2 = Student("1882845", "Mary", "Wilson")

print(repr(student2))
```

This method gives us the code to construct the string object

```
Student('1882845', 'Mary', 'Wilson')
```

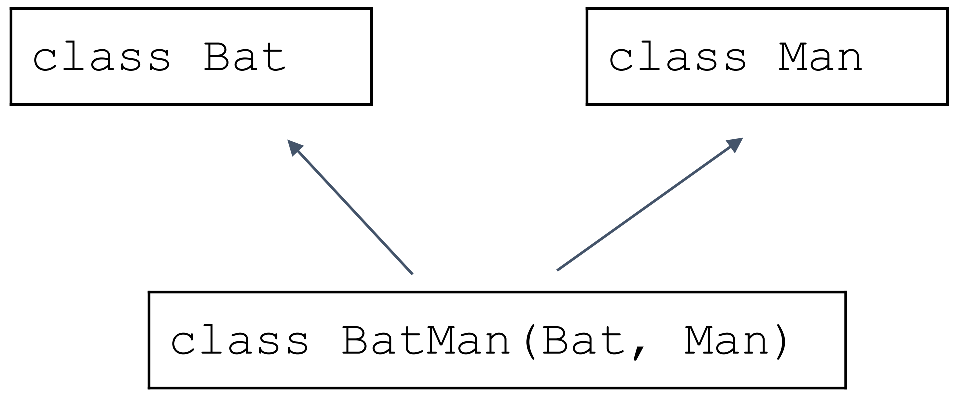
Help method

```
print(help(Student))
```

```
class Student(builtins.object)
 |  Class Student represents a student
 |  with the following attributes:
 |    id: student number
 |    first_name: first name
 |    last_name: last name
 |    username: Unix account username
 |
 |  Methods defined here:
 |
 |  __init__(self, id, first_name, last_name)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __repr__(self)
 |      Return repr(self).
 |
 |  __str__(self)
 |      Return str(self).
 |
 |  email(self)
 |      Get student's email: username@domain
 |
 |  email_alias(self)
 |      Get student's friendly-looking email:
 |      firstname.lastname.3IDdigits@domain
 |
 |  fullname(self)
 |      Get student's full name
 |
 |  home_dir(self)
 |      Get student's Unix home directory: studentDir/username
 |
 |  print_detail(self)
 |      Display student detail
 |
 |  ----------------------------------------------------------------------
```

```
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  email_domain = 'solla.sollew.edu'
 |
 |  student_dir = '/user/student'
```

Class inheritance



Python supports **multiple** class inheritance: a child class can inherit from multiple parent classes.

Class inheritance allow child class:

- To inherit all parent attributes and methods;
- To override parent attributes;
- To override parent methods.

Example

Consider a fictional Solla Sollew University again:

Each postgraduate student

- must register a thesis title

- is given a Unix home directory
  - but in a graduate directory
  - E.g. (`Adrian Creedon’s (0945720)` home directory is `/user/gradstudent/ac094`, instead of `/user/student/ac094`

- is given a home page
  - E.g. `Adrian Creedon’s` home page is `www.solla.sollew.edu/ac094`

Defining inheritance

```
class PostGradStudent(Student):
  """
  Class PostGradStudent represents a postgraduate student
  """

  student_dir = "/user/poststudent"

  def __init__(self, id, first_name, last_name, thesis):
    # calling parent class constructor
    super().__init__(id, first_name, last_name)

    # initialize thesis title
    self.thesis = thesis
```

Inheriting the class Student

overriding class attributes

Initialising inherited class attributes

adding more object attributes

```
# creating 3 postgraduate student objects
pg_student1 = PostGradStudent("0945720", "Adrian", "Creedon", "Polynomial Approximation of Functions")
pg_student2 = PostGradStudent("1892418", "Denis", "Carter", "Recursive array constructions")
pg_student3 = PostGradStudent("0793511", "Kara", "Kaufmann", "On Fundamental Semigroups")
```

Defining inheritance

```
# creating 3 postgraduate student objects
pg_student1 = PostGradStudent("0945720", "Adrian", "Creedon", "Polynomial Approximation of Functions")
pg_student2 = PostGradStudent("1892418", "Denis", "Carter", "Recursive array constructions")
pg_student3 = PostGradStudent("0793511", "Kara", "Kaufmann", "On Fundamental Semigroups")
```

```
# display object attributes

print(pg_student1.id)
print(pg_student1.first_name)
print(pg_student1.last_name)

print(pg_student1.thesis)
```

This is from parent class

This is from child class

Adding attribute and method

```
class PostGradStudent(Student):

  web_domain = "www.solla.sollew.edu"

  def web_address(self):
    """
    Get student's web address:
    webDomain/username
    """
    return PostGradStudent.web_domain + "/" + self.username
```

```
pg_student1 = PostGradStudent("0945720", "Adrian", "Creedon", "Polynomial Approximation of Functions")

print(pg_student1.web_address())

print(PostGradStudent.web_domain)
```

Overriding method

```python
class Student:
  student_dir = "/user/student"

  def home_dir(self):
    """
    Get student's Unix home directory:
    studentDir/username
    """
    return Student.student_dir + "/" + self.username
```

```python
class PostGradStudent(Student):
  student_dir = "/user/poststudent"

  def home_dir(self):
    """
    Get student's Unix home directory: studentDir/username
    Override the parent method with a new directory
    """
    return PostGradStudent.student_dir + "/" + self.username
```

Overriding method

```
# creating 3 student instances
student1 = Student("0973427", "John", "Smith")
student2 = Student("1882845", "Mary", "Wilson")
student3 = Student("0729032", "Ye", "Yang")
```

```
# creating 3 postgraduate student instances
pg_student1 = PostGradStudent("0945720", "Adrian", "Creedon", "Polynomial Approximation of Functions")
pg_student2 = PostGradStudent("1892418", "Denis", "Carter", "Recursive array constructions")
pg_student3 = PostGradStudent("0793511", "Kara", "Kaufmann", "On Fundamental Semigroups")
```

```
# compare the home directory between 2 students
print(student1.home_dir())

print(pg_student1.home_dir())
```

```
/user/student/js097
/user/poststudent/ac094
```

Overriding method

```
class PostGradStudent(Student):

  def print_detail(self):
    """
    Display student detail
    """
    super().print_detail()

    print("Thesis: " + self.thesis)
    print("Web address: " + self.web_address())
```

parent class

additional info from child class

```
Student ID: 0945720
First name: Adrian
Last name: Creedon
Full name: Adrian Creedon
Username: ac094
Email: ac094@solla.sollew.edu
Email alias: Adrian.Creedon.094@solla.sollew.edu
Home directory: /user/poststudent/ac094
Thesis: Polynomial Approximation of Functions
Web address: www.solla.sollew.edu/ac094
```

```
pg_student1 = PostGradStudent("0945720", "Adrian", "Creedon", "Polynomial Approximation of Functions")
pg_student1.print_detail()
```

Any questions?