TEXAS ADVANCED COMPUTING CENTER

WWW.TACC.UTEXAS.EDU

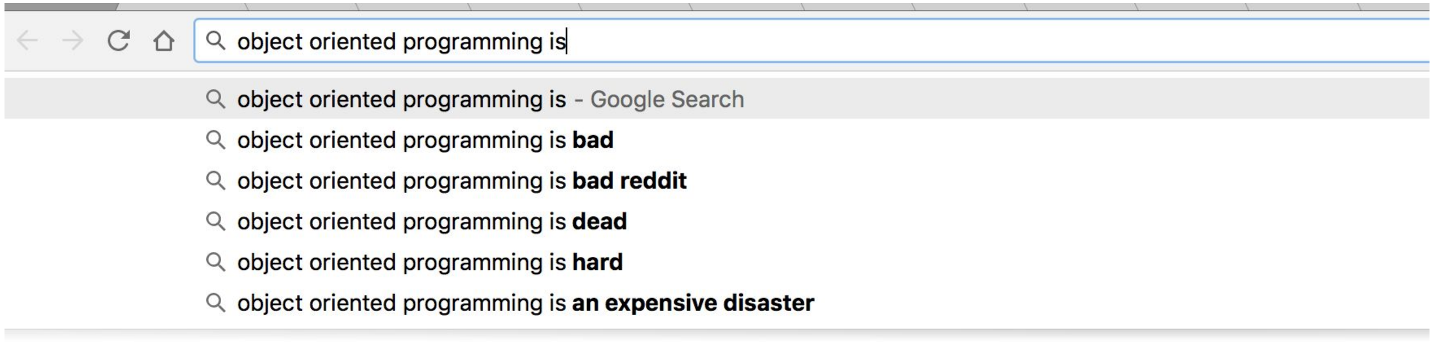
TEXAS
The University of Texas at Austin

Day 2: Object Oriented Python

S. Charlie Dey, Director of Training and Professional Development

Victor Eijkhout, Research Scientist

Introduction to Object Oriented Programming (OOP) - Don't listen to Google


object oriented programming is

object oriented programming is - Google Search

object oriented programming is **bad**

object oriented programming is **bad reddit**

object oriented programming is **dead**

object oriented programming is **hard**

object oriented programming is **an expensive disaster**

Sequential/Procedural Programming

the main focus of procedural programming is the "procedure" or the sequence of actions required to take input, process it, and produce output

Object Oriented Programming

Object-oriented programming is a programming paradigm based on the concept of "objects", defined by attributes (data); and methods (functions)

Object Oriented Programming

Objects may also *inherit* from other objects…

Sometimes even from *abstract* objects...

Object Oriented Programming - Definitions

Object: A set of data grouped together by a common theme.

Class: Describes the properties and methods of an object (it is like a blueprint or definition of an object). Properties can be variables, arrays, or data; Methods are functions that create/define behaviors of the object.

Property or Attribute: an embedded variable defined in an Object that stores descriptive data

Method: an embedded function in an Object that stores or generates behavioral data

Object Oriented Programming - Instantiation

Instantiation: creates new *instance* of an object to make the properties and methods accessible In other words, an object is an instance of a class.

Our first Class

The Point Class.

Points are defined by an x coordinate and a y coordinate

Our first Class

```
class Point(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y
```

To *Instantiate* `Point, p :`

```
 p = Point(5, 5) ## Instantiate a point at x=5, y=5
```

Our first Class, default values

```
class Point(object):
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
```

To *Instantiate* `Point, p :`

```
 p = Point() ## Instantiate a point at default x=0, y=0
```

Our first Class, self?

```
class Point(object):
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
```

Self is current **instance** of the object...

Adding Methods, Instance Methods

Let's add a method that calculates the distance from the origin:

Distance of a point P(x, y) from the origin is given by

d(0, P) = sqrt(px^2 + y^2)

in Python:

```
distance = np.sqrt(self.x**2 + self.y**2)
```

Adding Methods, Instance Methods

Let's add a method that calculates the distance from another point:

Distance of a point P(x, y) from the P2(x,y) is given by

d(P2, P) = sqrt((P2x-Px)^2 + (P2y-Py)^2)

in Python:

```
distance(P2) = np.sqrt((Point2.x - self.x)**2
                    + (Point2.y - self.y)**2)
```

Adding Class Methods, @classmethod

```
class Point(object):
    …
    @classmethod
    def RandomPoint(cls):
        return cls(int(rnd.random()*10), int(rnd.random()*10))
```

`cls` is a reference to the class

`self` is a reference to the current **instance** of the object

Static Methods, @staticmethod

```
class Point(object):
    …
    @staticmethod
    def FindFarthestPoint(Point_List):
        ...
```

static methods are methods related to the object, but do not require a reference to the current class or instance

Useful tips

```
class Point(object):
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
    def __str__(self):
        return ("[" + str(self.x) + ", " + str(self.y) + "]")
```

`__str__` allows us to define a string representation of an object

Useful tips

with

```
    def __str__(self)
```

we can:

```
   p = Point(1,1)
   print (p)
```

Disease Propagation Modeling

Goal: build an explicit simulation

We will maintain an explicit description of all the people in a population, and track each of their status.

We will use a simple model where a person can be:

- sick: when they are sick, they can infect other people;
- susceptible: they are healthy, but can be infected;
- recovered: they have been sick, but no longer carry the disease, and can not be infected for a second time;
- inoculated: they are healthy, do not carry the disease, and can not be infected.

We're going to start simple: any sick person is infectious.

We always start with just one person infected. The program will then track the population from day to day, running indefinitely until none of the population is sick. Since there is no re-infection, the run will always end.

Disease Propagation Modeling

We start by writing code that models a single person.
The main methods serve to infect a person, and to track their state.
We need to have some methods for inspecting that state.
Let's say "Joe" is infected on day 4, and is infected for 6 days
The intended output will look something like:

```
On day 1, Joe is susceptible
On day 2, Joe is susceptible
On day 3, Joe is susceptible
On day 4, Joe is sick (6 day(s) to go)
On day 5, Joe is sick (5 day(s) to go)
On day 6, Joe is sick (4 day(s) to go)
On day 7, Joe is sick (3 day(s) to go)
On day 8, Joe is sick (2 day(s) to go)
On day 9, Joe is sick (1 day(s) to go)
On day 10, Joe is recovered
```

Disease Propagation Modeling

- Write a `Person` class with methods:
  - `status_string()` : returns a description of the person’s state;
  - `update()` : update the person’s status to the next day;
  - `infect(n)` : infect a person, with the disease to run for n days;
  - `is_stable()` : report whether the person has been sick and is recovered.
  - `is_inoculated()` : report whether the person has been inoculated and therefore cannot be sick

Disease Propagation Modeling

```
import random as rnd
joe = Person()
while (not joe.is_stable()):
    joe.update()
    bad_luck = rnd.random();
    if (bad_luck > .9):
        joe.infect(5)
    print("joe is" + joe.status_string());
```

Disease Propagation Modeling

```
class Person(object):
    def __init__(self):
        self.current_status = "susceptible"
        self.days_sick = 0
        self.inoculated = false
        self.stable = false
    def is_stable(self):
        return self.stable
    def is_inoculated(self):
        return self.inoculated
    def infect(self, days):
        self.days_sick = days
```

Disease Propagation Modeling

```
def infected(self, days):
    self.days_sick = 0

def update(self):
    ?????
```

Disease Propagation Modeling

You will then expand on your code:

- **you will build a Population, which will be a list People objects**
- **make on person sick**
- **you'll then code random interactions within the population, allowing the disease to spread**
- **keep the simulation running until everyone is stable**
    - **record the number of your population**
    - **percentage that was immunized at the beginning**
    - **number of days until the entire population was healthy**
- **begin immunizing a percentage of your population, and rerun**

TACC

Disease Propagation Modeling

This is a research project, so you'll need to run various datasets and save the data to files
analyze the runs, draw intelligent conclusions

Record how long the disease runs through the population with various population sizes

With a fixed number of contacts and probability of transmission, how is this number of function of the percentage that is vaccinated?

Investigate the matter of ‘herd immunity’: if enough people are vaccinated, then some people who are not vaccinated will still never get sick. Let’s say you want to have this probability over 95 percent. Investigate the percentage of inoculation that is needed for this as a function of the contagiousness of the disease.