Lecture 12, 24 September 2024

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming and Data Structures with Python

Abstract data types

Interface – External operations to update & retrieve information

Implementation – How the data is stored, how operations work on this data

List

create() l = create()

append(v)

insert(p,v)

⋮

1. Flexible array
2. Nested dictionary

⋮

Object Oriented Programming

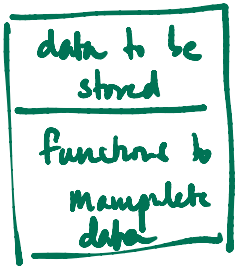
Traditionally

Focus on functions

Pass data to functions

O-O approach

Data + functionality is integrated



OO

```
def fn(a,b):
    ≡
```

template

Template for data + functions

Point

Data

Coordinates

xcoord
ycoord

Fns

translate $(\Delta x, \Delta y)$

distance()

display() ---

OO

Datatype template is called a **Class**

"Constructor"

```
Class Point:
    def __init__(self):
        self.xcoord = 0
        self.ycoord = 0
    def translate(self, deltax, deltay):
        self.xcoord = self.xcoord + deltax
        self.ycoord = self.ycoord + deltay
        return
```

How do xcoord & ycoord come into existence?

```
p = Point()
```

Creates a point

"object"

OO

An object is a concrete instance of a class

Realistically

```
p.setX(3)
p.setY(4)
```

```
p = Point
p.translate(-3,4)
```

self.xcoord = -3
self.ycoord = +4

```
p.translate(3,-4)
```

↳ (0,0)

Default arguments

int("7") ⇝ 7

int("AB") → Error

int("AB",16) ⇝ 171

def int(s, b=10):

default

int["7"] → b=10

```
def __init__(self, xval = 0, yval = 0):
    self.xcoord = xval
    self.ycoord = yval
```

```
p1 = Point(3,5)
        xcoord = 3
        ycoord = 5
p2 = Point()
        xcoord = 0
        ycoord = 0
p3 = Point(6) ⇝ (6,0)
```

(5) (5,)

`print(p1)` → calls a function to convert p1 into a string

```
def __str__():
    return ("(" + str(self.xcoord) + ","
                + str(self.ycoord) + ")")
```