

Lecture 13, 29 September 2026

Madhavan Mukund

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

Programming and Data Structures with Python

User defined datatypes

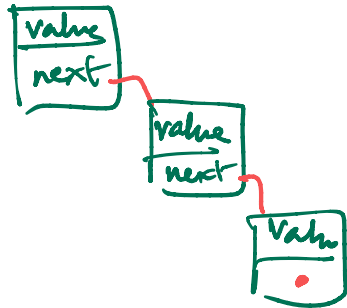


List

Flexibility ✓

No RAM ✗

Dictionary
Implementation



"Abstract" datatype

"Sequence" of values

→ Implementation

Operations

Create

Insert

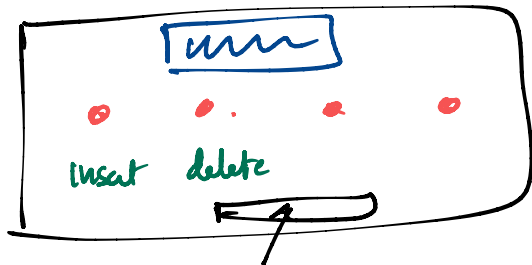
Delete

Reverse
Sort

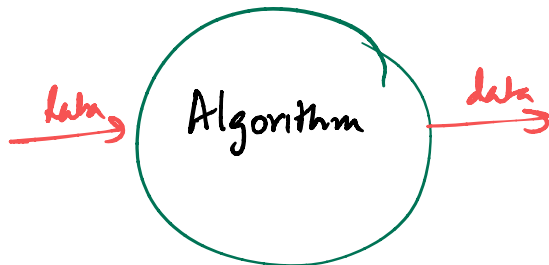
functional
specification
of how the
sequence
changes

How do store information?

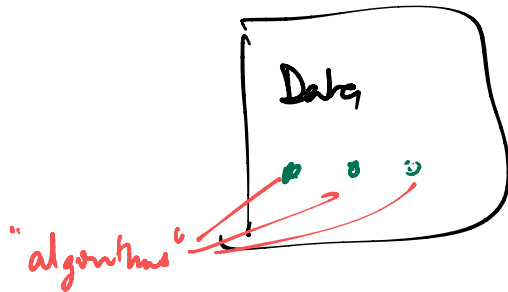
How do manipulate



1960s → 1970s

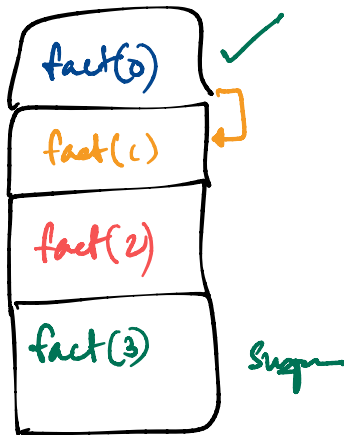


`l.append(v)`



Recursive function
fact(3) → fact(2)

"Activation
records"



"Stack"

Special type of seq.

push()

pop()

l.pop() in Python

Don't trust programmer discipline !

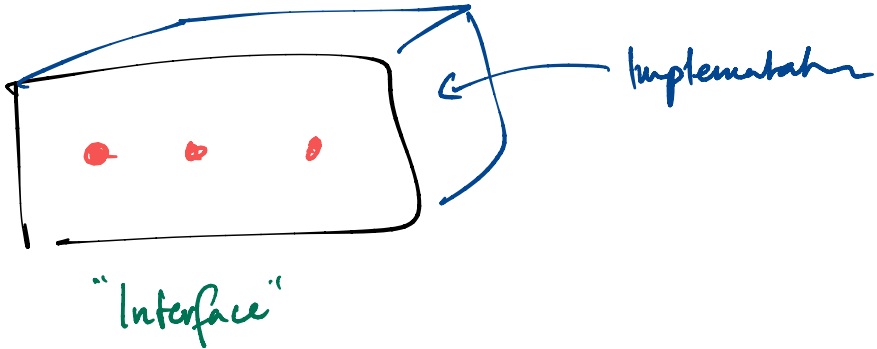
"Abstract" specifications

$$(s.\text{push}(v)).\text{pop}() \rightarrow v$$

We will not get into this !

Separation of interface & implementation

"Object Oriented" programming



Programming

Describe a template
for a data structure



Instantiate template
into concrete
data values

class

object

Example: Points in 2D

Information? Position (x, y)

Function?

"Move"

Translate $\Delta x, \Delta y$

Rotate θ

"Info"

Distance from $(0, 0)$

class Point:

 Set up the internal data

def translate — :

def rotate -- :

def odistance():

Setting up a Point

class Point:

Constructor

def __init__(self, newx, newy):

self.x = newx

self.y = newy

trigger
← some
action

p = Point(2, 7)

↓
creates a new Point
object

sets x = 2

y = 7

l = [2, 3, 5]

l = List(3, 4, 5)

class Point:

def __init__(-):

def translate(self, deltax, deltay):

self.x = self.x + deltax

self.y = self.y + deltay

self.y += deltay

$v = v \text{ op } a$
 $v \text{ op } = a$

p = Point(2,7)

p.translate(-1,3)



self.x = 1

self.y = 10

```
def odistance(self):  
    return  $\sqrt{(self.x)^2 + (self.y)^2}$ 
```

```
p = Point(2,7)  
p.translate(-1,3)  
d = p.odistance()  
     $\Downarrow$   
 $\sqrt{1^2 + 10^2} = \sqrt{101}$ 
```

Displaying objects?

`print(-)` $\leadsto$ displays `str(-)`

Special function `--str--`

```
def --str--(self):
```

```
    return "(" + str(self.x) + "," + str(self.y) + ")"
```

Default location for a new point $\rightarrow$ say $(0,0)$

def __init__(self, ~~newx, newy~~
newx=0, newy=0):

$p = \text{Point}(a,b) \leadsto \text{at } (a,b)$

$p = \text{Point}() \leadsto \text{at } (0,0)$

`int("7")` $\rightarrow$ 7

`int("hello")` $\rightarrow$ Error

`int("A0")` x

`int("A0", 16)` ✓

`def int(v, b=10):`