Lists

TEAM INFDEV

Hogeschool Rotterdam
Rotterdam, Netherlands

Introduction

Introduction

Lecture topics

- We now begin discussing specific, useful data structures
- These are already well known and understood
- Perfect for learning how a data structure is designed
- We begin with lists

Problem discussion

Problem discussion

Introduction

- So far we have been dealing with a single date in every variable
- For example, integer `0` in variable `i`
- Sometimes we need to store multiple things in the same variable

Problem discussion

Examples

- All players
- All the employees of the company
- All the trucks on the road
- All the aliens in the spaceship
- All the alien spaceships in the fleet
- ...

With variables?

```
truck1 = Truck(...)
truck2 = Truck(...)
...
truck10 = Truck(...)
```

Examples

With variables?

```
truck1 = Truck(...)
truck2 = Truck(...)
...
truck10 = Truck(...)
```

Examples

- Does this work?
- What if we have more or less than 10 trucks?

General idea

General idea

Introduction

- To solve this problem, we want to have all the data in a single variable
- The variable contains thus an **unknown** number of values
  - Might be empty
  - Might have only one element
  - Might have hundreds of elements
  - ...

General idea

Description

- To solve the issue, we will define an open-ended data structure
- The list is built as a linear chain of **nodes**
- In the simplest implementation, each node has
  - a **value**
  - a reference to the **next** elements
- We never really know how many elements we have in the list until we follow all the references through
- A special case is the empty list, which has no element and no reference to the next elements

```
+---+---+    +---+---+    +---+---+    +---+---+
| 3 | ------>| 7 | ------>| 4 | ------>|   |   |
+---+---+    +---+---+    +---+---+    +---+---+
```

Description

- Consider a list with elements 3, 7, and 4
- We need four nodes (the last is empty), all referencing the next

General idea

Description

- A list of values is built as either of:
  - An empty list `Empty`
  - A non-empty list containing the current value `v` and the rest of the list `vtail` `Node(v,tail)`
- **A list with three integers would be?**

General idea

Description

- A list of values is built as either of:
  - An empty list `Empty`
  - A non-empty list containing the current value `v` and the rest of the list `vtail` `Node(v,tail)`
- **A list with three integers would be?** `Node(1,Node(2,Node(3,Empty)))`
- **A list with two integers would be?**

General idea

Description

- A list of values is built as either of:
  - An empty list `Empty`
  - A non-empty list containing the current value `v` and the rest of the list `vtail` `Node(v,tail)`
- **A list with three integers would be?**
  `Node(1,Node(2,Node(3,Empty)))`
- **A list with two integers would be?**
  `Node(1,Node(2,Empty))`
- **An empty list would be?**

General idea

Description

- A list of values is built as either of:
  - An empty list `Empty`
  - A non-empty list containing the current value `v` and the rest of the list `vtail` `Node(v,tail)`
- **A list with three integers would be?** `Node(1,Node(2,Node(3,Empty)))`
- **A list with two integers would be?** `Node(1,Node(2,Empty))`
- **An empty list would be?** `Empty`

General idea

Description

- A list of values offers us three pieces of information:
  - A boolean `IsEmpty` indicating whether or not the list is empty
  - The value `Value` of the current element of the list in case it is **not empty**
  - The rest `Tail` of the list in case it is **not empty**
- Given a list `x`
  - **We can check if it is empty with?**

General idea

Description

- A list of values offers us three pieces of information:
  - A boolean `IsEmpty` indicating whether or not the list is empty
  - The value `Value` of the current element of the list in case it is **not empty**
  - The rest `Tail` of the list in case it is **not empty**
- Given a list `x`
  - **We can check if it is empty with?** `x.IsEmpty`
  - **We can read print its first value with?**

General idea

Description

- A list of values offers us three pieces of information:
  - A boolean `IsEmpty` indicating whether or not the list is empty
  - The value `Value` of the current element of the list in case it is **not empty**
  - The rest `Tail` of the list in case it is **not empty**
- Given a list `x`
  - **We can check if it is empty with?** `x.IsEmpty`
  - **We can read print its first value with?** `x.Value`
  - **We can print its second value with?**

General idea

Description

- A list of values offers us three pieces of information:
  - A boolean `IsEmpty` indicating whether or not the list is empty
  - The value `Value` of the current element of the list in case it is **not empty**
  - The rest `Tail` of the list in case it is **not empty**
- Given a list `x`
  - **We can check if it is empty with?** `x.IsEmpty`
  - **We can read print its first value with?** `x.Value`
  - **We can print its second value with?** `x.Tail.Value`
  - **We can print its third value with?**

General idea

Description

- A list of values offers us three pieces of information:
  - A boolean `IsEmpty` indicating whether or not the list is empty
  - The value `Value` of the current element of the list in case it is **not empty**
  - The rest `Tail` of the list in case it is **not empty**
- Given a list `x`
  - **We can check if it is empty with?** `x.IsEmpty`
  - **We can read print its first value with?** `x.Value`
  - **We can print its second value with?** `x.Tail.Value`
  - **We can print its third value with?** `x.Tail.Tail.Value`
  - ...

Technical details

Technical details

Introduction

- How is this done in Python?
- We shall build two data structures that, together, make up arbitrary lists
- We begin with the blueprints

The blueprint (**THIS IS NOT CODE!**)

```
Abstraction Empty =
  IsEmpty, which is always true

Abstraction Node =
  IsEmpty, which is always false
  Value, which contains the data of this element of the list
  Tail, which contains the remaining nodes of the list
```

Introduction

The blueprint (**THIS IS NOT CODE!**)

```
Abstraction Empty =
  IsEmpty, which is always true

Abstraction Node =
  IsEmpty, which is always false
  Value, which contains the data of this element of the list
  Tail, which contains the remaining nodes of the list
```

Introduction

- How do we translate this to Python?

The blueprint (**THIS IS NOT CODE!**)

```
Abstraction Empty =
  IsEmpty, which is always true

Abstraction Node =
  IsEmpty, which is always false
  Value, which contains the data of this element of the list
  Tail, which contains the remaining nodes of the list
```

Introduction

- How do we translate this to Python?
- Each abstraction becomes a class
- Each field is assigned under `__init__` to `self`

The actual code

```
class Empty:
  def __init__(self):
      self.IsEmpty = True
Empty = Empty()

class Node:
  def __init__(self, value, tail):
      self.IsEmpty = False
      self.Value   = value
      self.Tail    = tail
```

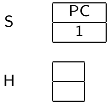
Note: we are switching to Python 3!

Technical details

Examples of list usage

- We now wish to build a list with our data structures
- We will build a list based on the input of the user
- User specifies how many, and which elements must go in the list

Examples of list usage



```
l = Empty
count = int(input("How␣many␣elements?"))
for i in range(0, count):
  v = int(input("Insert␣the␣next␣element")
  l = Node(v, l)
```

Examples of list usage

S

PC
1

H


```
1  l = Empty
2  count = int(input("How␣many␣elements?"))
3  for i in range(0, count):
4    v = int(input("Insert␣the␣next␣element")
5    l = Node(v, l)
```

S

PC	l	count	i	v
5	ref(0)	5	0	80085

H

0
[IsEmpty ↦ True]

Examples of list usage

S

PC	l	count	i	v
5	ref(0)	5	0	80085

H

0
[IsEmpty ↦ True]

```
l = Empty
count = int(input("How␣many␣elements?"))
for i in range(0, count):
  v = int(input("Insert␣the␣next␣element")
  l = Node(v, l)
```

Examples of list usage

S

PC	l	count	i	v
5	ref(0)	5	0	80085

H

0
[IsEmpty ↦ True]

```
1 l = Empty
2 count = int(input("How␣many␣elements?"))
3 for i in range(0, count):
4   v = int(input("Insert␣the␣next␣element")
5   l = Node(v, l)
```

S

PC	l	count	i	v
3	ref(1)	5	0	80085

H

0	1
[IsEmpty ↦ True]	[IsEmpty ↦ False; Value ↦ 80085; Tail ↦ ref(0)]

Examples of list usage

S

PC	l	count	i	v
5	ref(1)	5	1	8078

H

0	1
...	[IsEmpty ↦ False; Value ↦ 80085; Tail ↦ ref(0)]

```
1  l = Empty
2  count = int(input("How␣many␣elements?"))
3  for i in range(0, count):
4    v = int(input("Insert␣the␣next␣element")
5    l = Node(v, l)
```

Examples of list usage

S

PC	l	count	i	v
5	ref(1)	5	1	8078

H

0	1
…	[IsEmpty ↦ False; Value ↦ 80085; Tail ↦ ref(0)]

```
l = Empty
count = int(input("How␣many␣elements?"))
for i in range(0, count):
  v = int(input("Insert␣the␣next␣element")
  l = Node(v, l)
```

S

PC	l	count	i	v
5	ref(1)	5	1	8078

H

0	1	2
…	…	[IsEmpty ↦ False; Value ↦ 8078; Tail ↦ ref(1)]

Technical details

Examples of list usage

- We now wish to use the list we just built
- Specifically, we will print all its elements
- **How many elements does it have?**

Technical details

Examples of list usage

- We now wish to use the list we just built
- Specifically, we will print all its elements
- **How many elements does it have?**
- Unknown: it is specified by the user!

Examples of list usage

S

PC	I
1	ref(2)

H

0	1	2
[E $\mapsto$ T]	[E $\mapsto$ F; V $\mapsto$ 2; T $\mapsto$ ref(0)]	[E $\mapsto$ F; V $\mapsto$ 3; T $\mapsto$ ref(1)]

```
x = l
while not(x.IsEmpty):
  print(x.Value)
  x = x.Tail
```

Examples of list usage

S

PC	l
1	ref(2)

H

0	1	2
[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

```
x = l
while not(x.IsEmpty):
  print(x.Value)
  x = x.Tail
```

S

PC	l	x
2	ref(2)	ref(2)

H

0	1	2
[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

Examples of list usage

S

PC	l	x
2	ref(2)	ref(2)

H

0	1	2
[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

```
x = l
while not(x.IsEmpty):
  print(x.Value)
  x = x.Tail
```

What gets printed?

Examples of list usage

S

PC	l	x
2	ref(2)	ref(2)

H

0	1	2
[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

```
x = l
while not(x.IsEmpty):
  print(x.Value)
  x = x.Tail
```

What gets printed? `H[x][Value] = H[2][Value] = 3`

Examples of list usage

S

PC	l	x
2	ref(2)	ref(2)

H

0	1	2
[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

```
1  x = l
2  while not(x.IsEmpty):
3    print(x.Value)
4    x = x.Tail
```

What gets printed? `H[x][Value] = H[2][Value] = 3`

S

PC	l	x
3	ref(2)	ref(2)

H

0	1	2
[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

Examples of list usage

	PC	l	x
S	3	ref(2)	ref(2)

	0	1	2
H	[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

```
x = l
while not(x.IsEmpty):
  print(x.Value)
  x = x.Tail
```

Where is x.Tail?

Examples of list usage

Lists

TEAM INFDEV

S

PC	l	x
3	ref(2)	ref(2)

H

0	1	2
[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

```
1  x = l
2  while not(x.IsEmpty):
3    print(x.Value)
4    x = x.Tail
```

Where is x.Tail? `H[x][Tail] = H[2][Tail] = ref(1)`

Examples of list usage

S

PC	l	x
3	ref(2)	ref(2)

H

0	1	2
[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

```
1  x = l
2  while not(x.IsEmpty):
3    print(x.Value)
4    x = x.Tail
```

Where is x.Tail? `H[x][Tail] = H[2][Tail] = ref(1)`

S

PC	l	x
4	ref(2)	ref(1)

H

0	1	2
[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

Examples of list usage

S

PC	l	x
3	ref(2)	ref(1)

H

0	1	2
[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

```
x = l
while not(x.IsEmpty):
  print(x.Value)
  x = x.Tail
```

Where is x.Tail?

Examples of list usage

S

PC	l	x
3	ref(2)	ref(1)

H

0	1	2
[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

```
1  x = l
2  while not(x.IsEmpty):
3    print(x.Value)
4    x = x.Tail
```

Where is x.Tail? `H[x][Tail] = H[1][Tail] = ref(0)`

Examples of list usage

	PC	l	x
S	3	ref(2)	ref(1)

	0	1	2
H	[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

```
1  x = l
2  while not(x.IsEmpty):
3    print(x.Value)
4    x = x.Tail
```

Where is x.Tail? `H[x][Tail] = H[1][Tail] = ref(0)`

	PC	l	x
S	4	ref(2)	ref(0)

	0	1	2
H	[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

Examples of list usage

S

PC	l	x
2	ref(2)	ref(0)

H

0	1	2
[E $\mapsto$ T]	[E $\mapsto$ F; V $\mapsto$ 2; T $\mapsto$ ref(0)]	[E $\mapsto$ F; V $\mapsto$ 3; T $\mapsto$ ref(1)]

```
1  x = l
2  while not(x.IsEmpty):
3    print(x.Value)
4    x = x.Tail
```

What is the value of x.IsEmpty?

Examples of list usage

S

PC	l	x
2	ref(2)	ref(0)

H

0	1	2
[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

```
x = l
while not(x.IsEmpty):
  print(x.Value)
  x = x.Tail
```

What is the value of x.IsEmpty? `H[x][IsEmpty] = H[0][IsEmpty] = True`

Examples of list usage

S

PC	l	x
2	ref(2)	ref(0)

H

0	1	2
[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

```
x = l
while not(x.IsEmpty):
  print(x.Value)
  x = x.Tail
```

What is the value of x.IsEmpty? `H[x][IsEmpty] = H[0][IsEmpty] = True`

S

PC	l	x
5	ref(2)	ref(0)

H

0	1	2
[E ↦ T]	[E ↦ F; V ↦ 2; T ↦ ref(0)]	[E ↦ F; V ↦ 3; T ↦ ref(1)]

In-class homework

In-class homework

Implement the following (on paper)

- Read a list from the user input
- Remove all odd numbers
- A “volunteer” runs the steps on paper with the memory model

In-class homework

Implement the following (on paper)

- Read a list from the user input
- Sum all its values
- A “volunteer” runs the steps on paper with the memory model

In-class homework

Implement the following (on paper)

- Read a list from the user input
- Reverse it
- A “volunteer” runs the steps on paper with the memory model

In-class homework

Implement the following (on paper)

- Read two lists from the user input
- Append the second to the first (concatenate them)
- A “volunteer” runs the steps on paper with the memory model

Conclusion

Conclusion

Lecture topics

- What we solved today was the issue of representing multiple data inside a single variable
- We used a simple data structure, the **list**
- We showed how we can consume (use) the list through looping

This is it!

The best of luck, and thanks for the attention!