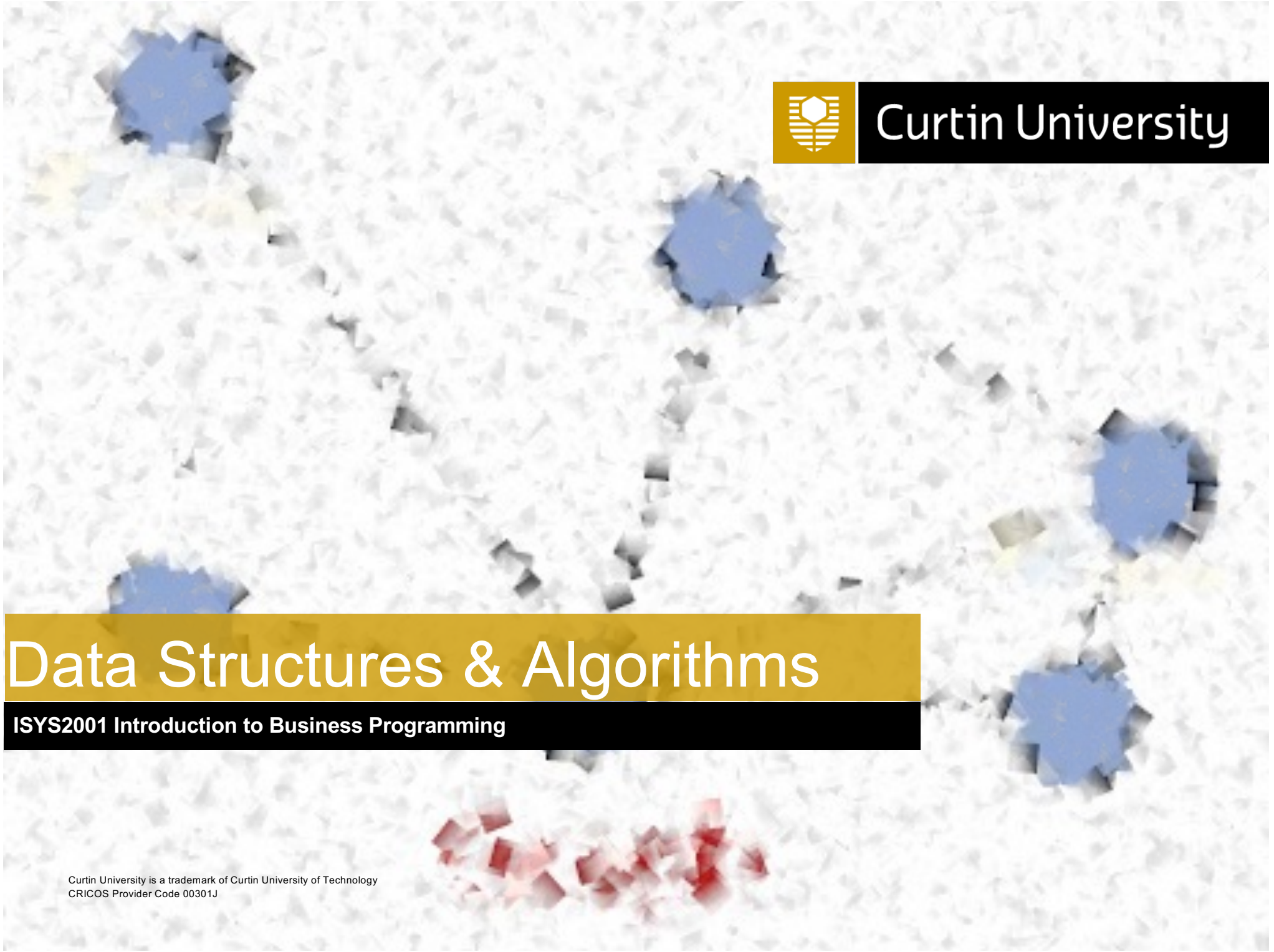


Curtin University

Data Structures & Algorithms

ISYS2001 Introduction to Business Programming

ELECTRONIC WARNING NOTICE FOR COPYRIGHT STATUTORY LICENCES

Curtin University

I acknowledge the traditional custodians of the land on which I work and live, and recognise their continuing connection to land, water and community. I pay respect to elders past, present and emerging.

Today

- What are Data Structures?
- Basic Data Structures in Python
- What are Algorithms?
- Important Algorithms in Python
- Algorithmn Examples

What are data Structures?

Python Data Structures

- Built-in
  - Lists
  - Tuples and Named Tuples
  - Sets
  - Dictionaries
- User Defined (using built-in)
  - Stacks
  - Queues
  - Linked Lists
  - Hash Maps
  - Trees
  - Graphs

Lists

```
[4]: list1 = []  # creating an empty list
     list2 = [1, 2.4, 'abc', (1, 2), [6, 78, 9]]  # creating a list of different values

     print("The type of", list1, "is:", type(list1))

     print("The type of", list2, "is:", type(list2))
```

```
The type of [] is: <class 'list'>
The type of [1, 2.4, 'abc', (1, 2), [6, 78, 9]] is: <class 'list'>
```

Lists

[5]:
```
list_1 = [1, 3, 5.6, 'a', "Hello World", 0.4]

print("list_1[2:5] =", list_1[2:5])  # slicing 3rd to 5th element

print("list_1[-5:-2] =", list_1[-5:-2])  # slicing last 5th to 3rd element

print("list_1[3:] =", list_1[3:])  # slicing all elements after 3rd one

print("list_1[:2] =", list_1[:2])  # slicing all elements till 2nd one

print("list_1[:-2] =", list_1[:-2])  # slicing all elements till last 2nd one
```

```
list_1[2:5] = [5.6, 'a', 'Hello World']
list_1[-5:-2] = [3, 5.6, 'a']
list_1[3:] = ['a', 'Hello World', 0.4]
list_1[:2] = [1, 3]
list_1[:-2] = [1, 3, 5.6, 'a']
```

Lists

- my_list.append(x)
- my_list.extend(L)
- my_list.insert(i, x)
- my_list.remove(x)
- my_list.pop([i])
- my_list.clear()
- my_list.index(x)
- my_list.count(x)
- my_list.sort()
- my_list.reverse()
- my_list.copy

Tuples

```
[6]: my_tuple = (91, 3.4, "hello")

print(my_tuple[0])

my_tuple[0] = 12
```

```
91
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [6], in <cell line: 5>()
      1 my_tuple = (91, 3.4, "hello")
      3 print(my_tuple[0])
----> 5 my_tuple[0] = 12

TypeError: 'tuple' object does not support item assignment
```

Tuples - unpacking

[7]:
```
tup = 1, 2, 3  # packing, parethenisis optional

print("The type of ", tup, "is: ", type(tup))

a, b, c = tup  #unpacking

print(a)
print(b)
print(c)
```

```
The type of  (1, 2, 3) is:  <class 'tuple'>
1
2
3
```

Sets

[11]:
```
set1 = {1, 2, 'a', 't', 7.5}
set2 = {'g', 2, 'u', 5.6}

set1.add(5)
print(set1)
print(len(set1))

set2.pop()
print(set2)

set1.remove('a')
print(set1)
set1.discard(1)
print(set1)

set2.remove(5.6)

set1.discard(5)
print(set1)

set2.clear()
print(len(set2))
```

```
{1, 2, 5, 7.5, 't', 'a'}
6
{'g', 'u', 5.6}
{1, 2, 5, 7.5, 't'}
{2, 5, 7.5, 't'}
{2, 7.5, 't'}
0
```

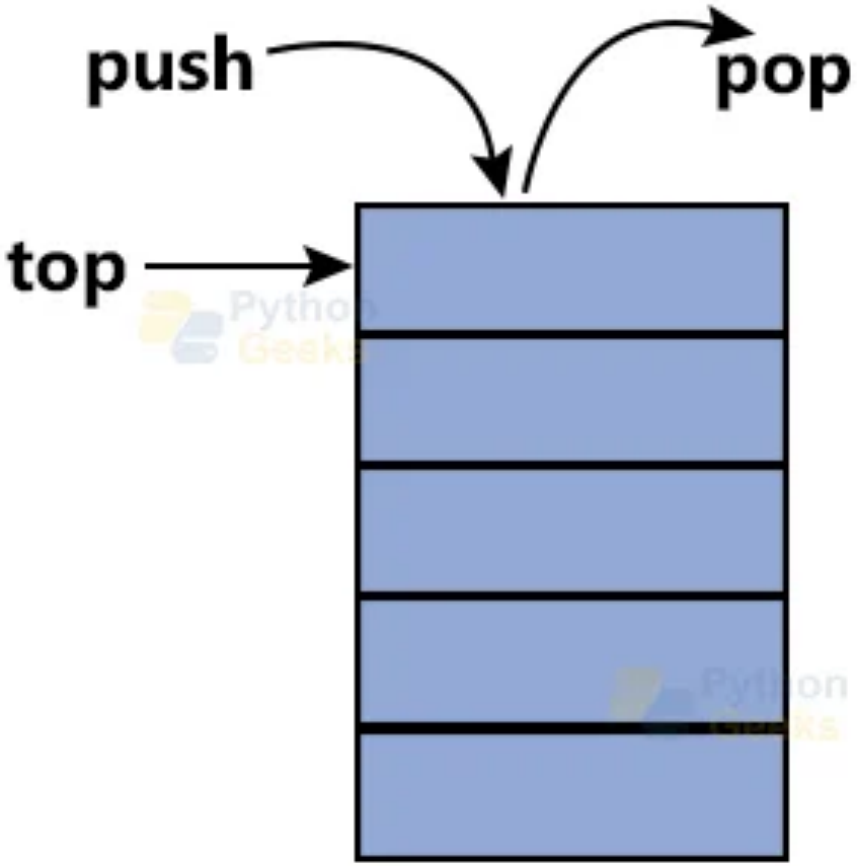
Dictionaries – key:value pairs

[12]:
```
phonebook = {
    "bob": 7387,
    "alice": 3719,
    "jack": 7052,
}

squares = {x: x * x for x in range(6)}

print(phonebook["alice"])

squares
```

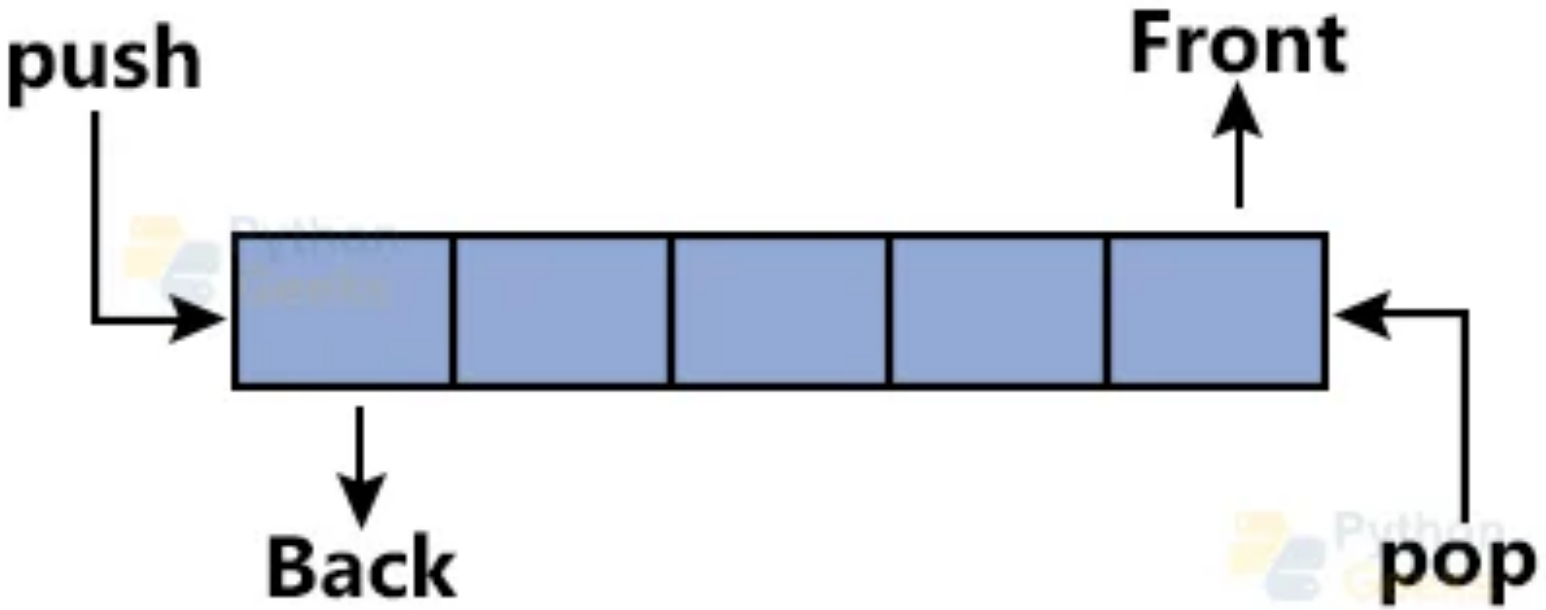
```
3719
```

[12]:
```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

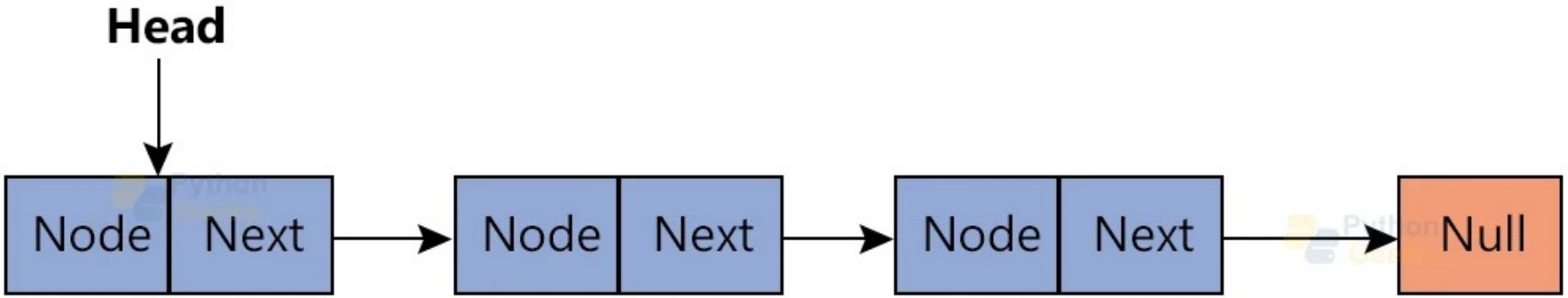
Stacks



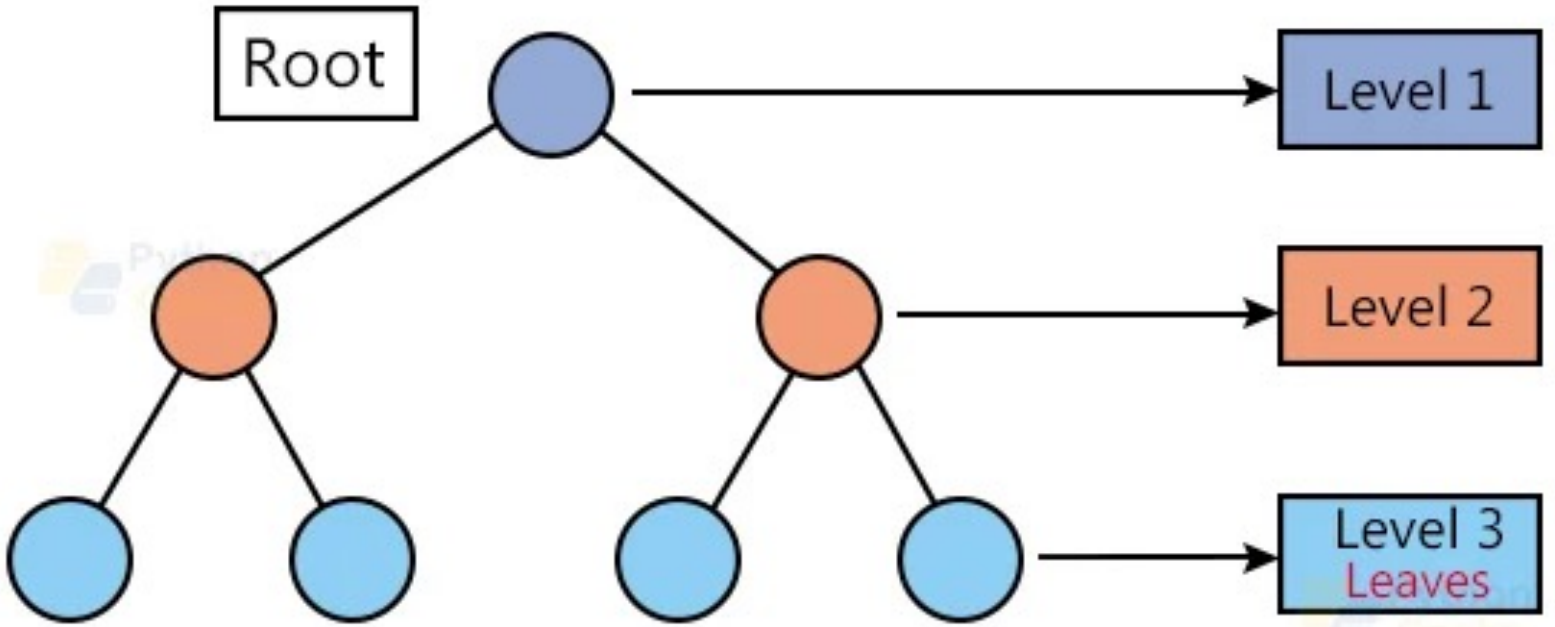
Queues



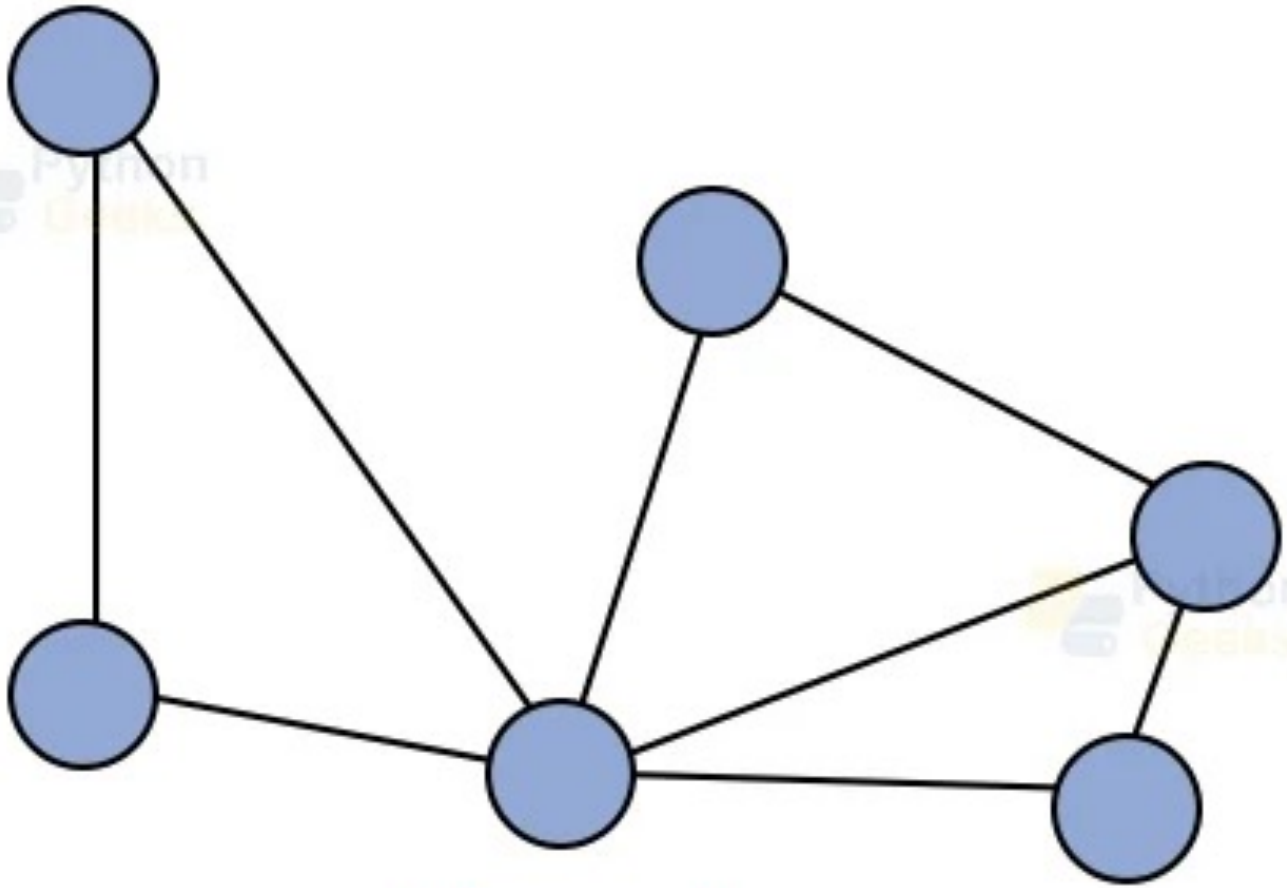
Linked Lists



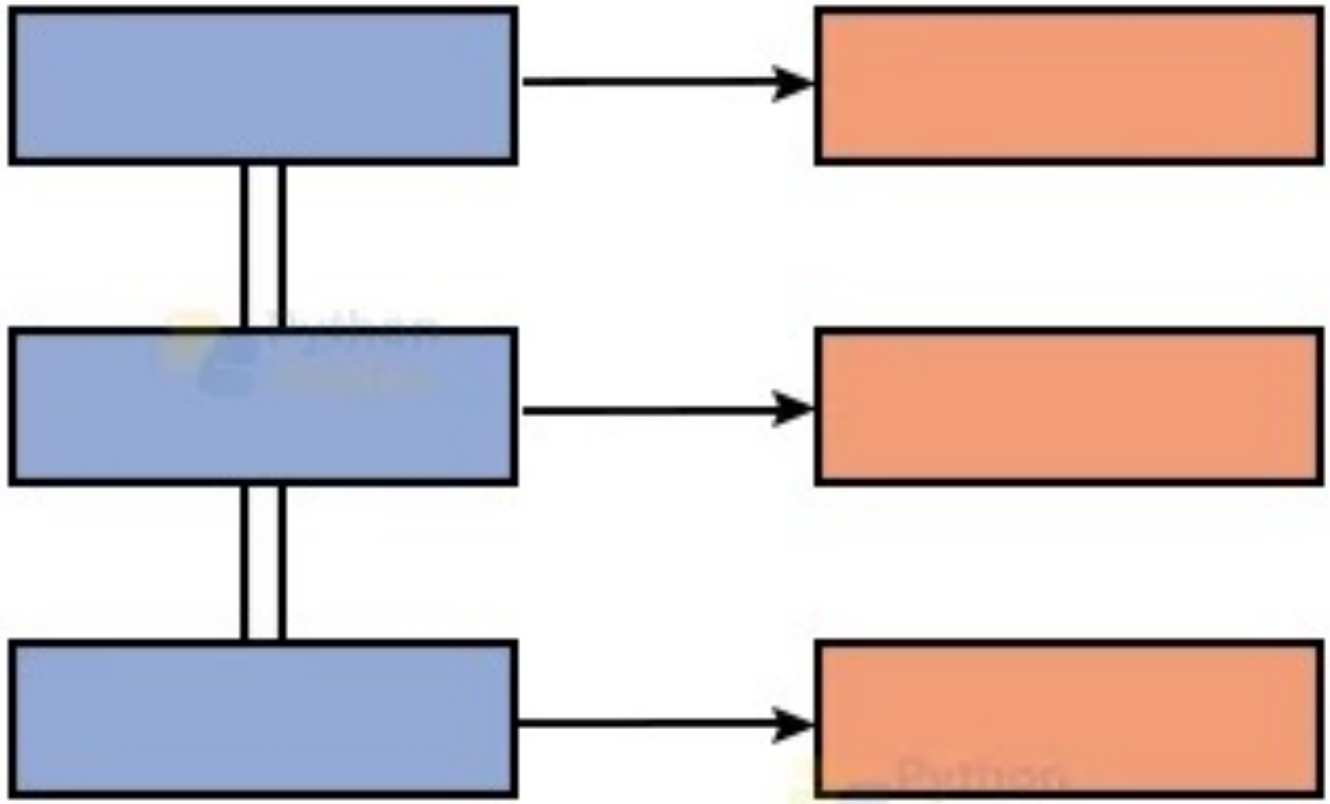
Tree



Graph



Hash Maps



What are Algoritms?

Fundamental Algorithms

- Searching
- Sorting
- Graph
- Artificial Intelligence
- Dynamic Programming
- Greedy Algorithms
- Divide and Conquer
- Backtracking
- Numerical Algorithms

How Compare

- Measure Space (memory, disk space)
- Measure Time (number of operations)
- Use Big-O notation
- O(1) – constant

  O(log n) – logarithmic

  O(n) – linear

  O(n^2) – Quadratic

  O(2^n) - Exponential

Example – Linear Search

- Find an element in a list by iterating through the list.
- Time Complexity: O(n)

```
def linear_search(arr, target):
    for index, element in enumerate(arr):
        if element == target:
            return index  # Target found, return its index

    return -1  # Target not found, return -1
```

Example – Binary Search

- Search a ***sorted*** list by repeatedly dividing the list in half.
- Time Complexity: O(log n)

```python
def binary_search(arr, target):
    low, high = 0, len(arr) - 1

    while low <= high:
        mid = (low + high) // 2
        mid_value = arr[mid]

        if mid_value == target:
            return mid  # Target found, return its index
        elif mid_value < target:
            low = mid + 1
        else:
            high = mid - 1

    return -1  # Target not found, return -1
```

Example - Selection Sort

- Sort a list by repeatedly swapping adjacent elements if they are in the wrong order.
- Time Complexity: O(n^2)

```
def selection_sort(arr):
    n = len(arr)

    for i in range(n - 1):
        min_index = i
        for j in range(i + 1, n):
            if arr[j] < arr[min_index]:
                min_index = j
        arr[i], arr[min_index] = arr[min_index], arr[i]

    return arr
```

Summary

- Data structures are essential for organizing and managing data efficiently.
- Basic data structures in Python: Lists, Tuples, Sets, and Dictionaries.
- Algorithms are step-by-step procedures
- Study Algorithms improves performance