

The background is a solid pink color. It is decorated with various hand-drawn geometric shapes in white and black. These include a dashed line in the top left, a white triangle in the top center, a black zigzag line in the top right, a white circle in the top right, two parallel black lines in the top right, a white triangle in the top right, a large black circle in the bottom right, a white circle in the bottom right, a black plus sign in the bottom left, a white circle in the bottom left, a white triangle in the bottom left, and a black plus sign in the bottom left.

# Welcome!

We'll get started shortly ...



# CS 49 Section


Week 11

Surajit A Bose





# Agenda

- Logistics and check-ins
  - Review of lecture concepts
    - Dictionaries
    - [Dictionary practice](#)
  - Section Problem: [Find Grandchildren](#)
- 

# Logistics






# How to get hold of me / get help+

- The [section forum](#), 24 hr turnaround
- Email: [bozesurajit@fhda.edu](mailto:bozesurajit@fhda.edu), 24 hr turnaround
- Office hours:
  - ~~○ On campus: Tuesdays 12:00 noon to 1:30 pm, room 4218 in the STEM center. Entry is from room 4213. No in person hours during finals~~
  - By appointment on Zoom
- Other resources:
  - Contact Lane via Canvas
  - ~~○ [Online](#) or [in person](#) tutoring via the STEM center (Room 4213)~~





# Check In

- Any questions about:
    - Lists
    - Concepts from previous weeks
    - Any homework or EC problems
  - Please tell us in the chat what your favorite cuisine is! Thai, BBQ, Japanese, Mediterranean, etc.
- 




# Lecture Review: Dictionaries





# Introducing Dictionaries

- One container type in Python is dictionaries (**dict**)
  - A dictionary enables very fast lookups of a **value** via an associated **key**
  - For example, if the college needs to look up info about a student:
    - Looking up via the CWID will return information about the student's name, enrollment status, major, GPA, etc.
    - The CWID is the key, the student info is the value
  - Thus, a dictionary is a collection of **key : value** pairs
  - Conceptual examples:
    - Country as key, capital as value ('Canada' : 'Ottawa')
    - Hex value as key, color name as value (0xb70b2f : '**cardinal**')
- 



# Creating and Modifying Dictionaries

- To create an empty dictionary: `{}` or `dict()`  
`my_dict = {}`
- To create a new non-empty dictionary:  
`state_caps = {'Oregon' : 'Salem', 'Idaho': 'Boise',  
              'Hawaii' : 'Honolulu', 'Alaska' : 'Juneau'}`
- To add a key : value pair to the dictionary: `dict[key] = value`  
`state_caps['Maryland'] = 'Annapolis'`  
`state_caps['Kentucky'] = 'Lexington'`
- Oops! To update an existing value, use the same syntax:  
`state_caps['Kentucky'] = 'Frankfort'`
- This will replace the existing value for that key with the new value

# Accessing Data in Dictionaries +

- To access a value, use the key:

```
md_cap = state_caps['Maryland']  
# md_cap now has the value 'Annapolis'
```

- Trying to access the value for a non-existent key results in an error:

```
ca_cap = state_caps['California']    # KeyError
```

- To see if a particular key is in a dictionary, use **in**:

```
print('California' in state_caps)    # prints False  
print('Hawaii' in state_caps)        # prints True
```

- To see if a particular value is in a dictionary, use **in dict.values()**:

```
print('Boise' in state_caps.values()) # prints True
```

# Constraints on Dictionary Keys +

- Keys must be unique. Cannot have:

```
mixed_caps = {'Georgia' : 'Tbilisi',  
              'Georgia' : 'Atlanta'}    # 'Tbilisi' is replaced
```


- Keys must be immutable. Recall:
  - Atomic types (**int**, **Boolean**, **float**) are immutable
  - Of the container types, **str** is immutable
- Lists are mutable, so a **list** cannot be used as a key
- These constraints do not apply to values
  - Two different keys can have the same associated value
  - Values can be mutable types like lists or even other dictionaries

# Iterating over Dictionaries

- To iterate over all key : value pairs: `dict.items()`  
`for state, city in state_caps.items():`  
`print(f'The capital of {state} is {city}')`
- To iterate over just the keys:  
`for state in state_caps:`  
`print(state)`
- To iterate over just the values, use `dict.values()`:  
`for city in state_caps.values():`  
`print(city)`
- Likewise, there is also a `dict.keys()`



# Dictionaries vs Lists

- We've seen that data types can be:
    - atomic (**int**, **float**, **Boolean**, **None**) or container (**str**, **list**)
    - immutable (**str**, any atomic type) or mutable (**list**)
  - **dict** is a mutable container type, like **list**
  - Data types can also be ordered or unordered
    - **list** and **str** are ordered: data is consecutive by index
    - **list** and **str** indices are always integers
  - **dict** is unordered: the sequence is arbitrary
  - **dict** keys can be any immutable value (including integers, which could be non-consecutive)
- 

The background is a solid orange color. It is decorated with various hand-drawn geometric shapes in white and black. These include: a dashed line in the top left; a white triangle in the top center; a black zigzag line in the top right; a white circle in the top right; two parallel black lines in the top right; a white triangle in the top right; a large black circle on the right edge; a white triangle in the bottom left; a black plus sign in the bottom left; a white circle in the bottom center; a white triangle in the bottom center; a dashed line in the bottom center; a black plus sign in the bottom center; a black circle in the bottom center; a white circle in the bottom right; and a black circle in the bottom right.

**Any Questions?**



# Dictionaries Practice

<https://codeinplace.stanford.edu/foothill-cs49/ide/p/sy6eC9jGvP11LL9qp8dV>





# Dictionaries: Key-Value Pairs +

- I have three dogs. Rover is a lab, Spot a mutt, and Max a pug. Create a new dictionary **breeds** with the names as keys, the breeds as values.



# Dictionaries: Key-Value Pairs +

- I have three dogs. Rover is a lab, Spot a mutt, and Max a pug. Create a new dictionary **breeds** with the names as keys, the breeds as values.

```
breeds = {  
    'Rover' : 'lab',  
    'Spot' : 'mutt',  
    'Max' : 'pug'  
}
```



# Dictionaries: Key-Value Pairs +

- I have three dogs. Rover is a lab, Spot a mutt, and Max a pug. Create a new dictionary **breeds** with the names as keys, the breeds as values.

```
breeds = {  
    'Rover' : 'lab',  
    'Spot' : 'mutt',  
    'Max' : 'pug'  
}
```

- Fetch Rover's breed and store it in a variable called **rover\_breed**
- 

# Dictionaries: Key-Value Pairs

- I have three dogs. Rover is a lab, Spot a mutt, and Max a pug. Create a new dictionary **breeds** with the names as keys, the breeds as values.

```
breeds = {  
    'Rover' : 'lab',  
    'Spot' : 'mutt',  
    'Max' : 'pug'  
}
```

- Fetch Rover's breed and store it in a variable called **rover\_breed**  
**rover\_breed = breeds['Rover']**



# Dictionaries: Key-Value Pairs +

- I have a new dog! Her name is Fifi. She is also a mutt. Add Fifi to the **breeds** dictionary.



# Dictionaries: Key-Value Pairs +

- I have a new dog! Her name is Fifi. She is also a mutt. Add Fifi to the **breeds** dictionary.

```
breeds['Fifi'] = 'mutt'
```



# Dictionaries: Key-Value Pairs +

- I have a new dog! Her name is Fifi. She is also a mutt. Add Fifi to the **breeds** dictionary.

```
breeds['Fifi'] = 'mutt'
```

- Oops! Fifi is very upset. She says she is not a mutt, she is a poodle!
- 

# Dictionaries: Key-Value Pairs +

- I have a new dog! Her name is Fifi. She is also a mutt. Add Fifi to the **breeds** dictionary.

```
breeds['Fifi'] = 'mutt'
```

- Oops! Fifi is very upset. She says she is not a mutt, she is a poodle!

```
breeds['Fifi'] = 'poodle'
```



# Dictionaries: Key-Value Pairs +

- I have a new dog! Her name is Fifi. She is also a mutt. Add Fifi to the **breeds** dictionary.

```
breeds['Fifi'] = 'mutt'
```

- Oops! Fifi is very upset. She says she is not a mutt, she is a poodle!

```
breeds['Fifi'] = 'poodle'
```

- Hm, I can't remember whether Max is in the dictionary. Can you check?





# Dictionaries: Key-Value Pairs +

- I have a new dog! Her name is Fifi. She is also a mutt. Add Fifi to the **breeds** dictionary.

```
breeds['Fifi'] = 'mutt'
```

- Oops! Fifi is very upset. She says she is not a mutt, she is a poodle!

```
breeds['Fifi'] = 'poodle'
```

- Hm, I can't remember whether Max is in the dictionary. Can you check?

```
print('Max' in breeds)    # prints True
```

# Dictionaries: Key-Value Pairs +

- I have a new dog! Her name is Fifi. She is also a mutt. Add Fifi to the **breeds** dictionary.

```
breeds['Fifi'] = 'mutt'
```

- Oops! Fifi is very upset. She says she is not a mutt, she is a poodle!

```
breeds['Fifi'] = 'poodle'
```

- Hm, I can't remember whether Max is in the dictionary. Can you check?

```
print('Max' in breeds)    # prints True
```

- Print out all the key-value pairs.

# Dictionaries: Key-Value Pairs +

- I have a new dog! Her name is Fifi. She is also a mutt. Add Fifi to the **breeds** dictionary.

```
breeds['Fifi'] = 'mutt'
```

- Oops! Fifi is very upset. She says she is not a mutt, she is a poodle!

```
breeds['Fifi'] = 'poodle'
```

- Hm, I can't remember whether Max is in the dictionary. Can you check?

```
print('Max' in breeds)    # prints True
```

- Print out all the key-value pairs.

```
for dog, breed in breeds.items():  
    print(f'{dog} is a {breed}')
```



# Section problem: Find Grandchildren

<https://codeinplace.stanford.edu/foothill-cs49/ide/a/findgrandchildren>



# Find Grandchildren

+

- Given a **dict** with parents as keys, and lists of their children as values:  

```
parents_dict = { 'Khaled' : ['Chibundu', 'Jesmyn'],  
                'Daniel' : ['Khaled', 'Eve'],  
                'Jesmyn' : ['Frank'],  
                'Eve' : ['Grace'] }
```
- Create a **dict** with grandparents as keys, and grandchildren as values

# Find Grandchildren

- Given a **dict** with parents as keys, and lists of their children as values:

```
parents_dict = { 'Khaled' : ['Chibundu', 'Jesmyn'],  
                'Daniel' : ['Khaled', 'Eve'],  
                'Jesmyn' : ['Frank'],  
                'Eve': ['Grace'] }
```


- Create a **dict** with grandparents as keys, and grandchildren as values
- Only actual grandparents should be keys — i.e., no empty lists for values

```
parents_dict = {  
    'Jesmyn' : [],  
    'Eve': [] }  
}
```



# Find Grandchildren

- Given a **dict** with parents as keys, and lists of their children as values:  

```
parents_dict = { 'Khaled' : ['Chibundu', 'Jesmyn'],  
                'Daniel' : ['Khaled', 'Eve'],  
                'Jesmyn' : ['Frank'],  
                'Eve' : ['Grace'] }
```
  - Create a **dict** with grandparents as keys, and grandchildren as values
  - How should we identify the needed key : value pairs?
- 

# Find Grandchildren

+



- Given a **dict** with parents as keys, and lists of their children as values:  

```
parents_dict = { 'Khaled' : ['Chibundu', 'Jesmyn'],  
                'Daniel' : ['Khaled', 'Eve'],  
                'Jesmyn' : ['Frank'],  
                'Eve' : ['Grace'] }
```
- Create a **dict** with grandparents as keys, and grandchildren as values
- How should we identify the needed key : value pairs?
- For keys, consider:
  - Is Khaled a grandparent? How do we know?
  - How about Daniel? Jesmyn? Eve?





# Find Grandchildren

- Given a **dict** with parents as keys, and lists of their children as values:  

```
parents_dict = { 'Khaled' : ['Chibundu', 'Jesmyn'],  
                'Daniel' : ['Khaled', 'Eve'],  
                'Jesmyn' : ['Frank'],  
                'Eve' : ['Grace'] }
```
- Create a **dict** with grandparents as keys, and grandchildren as values
- How should we identify the needed key : value pairs?
- For keys, consider:
  - Is Khaled a grandparent? How do we know?
  - How about Daniel? Jesmyn? Eve?
- A parent **X** is a grandparent if any of their children is also a parent

# Find Grandchildren

+

- Given a **dict** with parents as keys, and lists of their children as values:  

```
parents_dict = { 'Khaled' : ['Chibundu', 'Jesmyn'],  
                'Daniel' : ['Khaled', 'Eve'],  
                'Jesmyn' : ['Frank'],  
                'Eve' : ['Grace'] }
```
- Create a **dict** with grandparents as keys, and grandchildren as values
- How should we identify the needed key : value pairs?
- A key **X** in **parents\_dict** should be a key in **grandparents\_dict** if any of the list elements in the value **parents\_dict[X]** is itself a key in **parents\_dict**

# Find Grandchildren

+


- Given a **dict** with parents as keys, and lists of their children as values:  

```
parents_dict = { 'Khaled' : ['Chibundu', 'Jesmyn'],  
                'Daniel' : ['Khaled', 'Eve'],  
                'Jesmyn' : ['Frank'],  
                'Eve' : ['Grace'] }
```
- Create a **dict** with grandparents as keys, and grandchildren as values
- How should we identify the needed key : value pairs?
- For values, consider:
  - Who are Khaled's grandchildren?
  - How about Daniel's?



# Find Grandchildren

- Given a **dict** with parents as keys, and lists of their children as values:  

```
parents_dict = { 'Khaled' : ['Chibundu', 'Jesmyn'],  
                'Daniel' : ['Khaled', 'Eve'],  
                'Jesmyn' : ['Frank'],  
                'Eve' : ['Grace'] }
```
  - Create a **dict** with grandparents as keys, and grandchildren as values
  - How should we identify the needed key : value pairs?
  - For values, consider:
    - Who are Khaled's grandchildren?
    - How about Daniel's?
  - X**'s grandchildren are *all* the children of *each* of their children
- 

# Find Grandchildren

- Given a **dict** with parents as keys, and lists of their children as values:  

```
parents_dict = { 'Khaled' : ['Chibundu', 'Jesmyn'],  
                'Daniel' : ['Khaled', 'Eve'],  
                'Jesmyn' : ['Frank'],  
                'Eve' : ['Grace'] }
```
- Create a **dict** with grandparents as keys, and grandchildren as values
- How should we identify the needed key : value pairs?
- To construct the value at **grandparents\_dict[X]**:
  - Identify the children in **parents\_dict[X]** who are themselves keys in **parents\_dict**
  - Combine their values in **parents\_dict** into a single list

# Find Grandchildren

- Given a **dict** with parents as keys, and lists of their children as values:

```
parents_dict = { 'Khaled' : ['Chibundu', 'Jesmyn'],  
                'Daniel' : ['Khaled', 'Eve'],  
                'Jesmyn' : ['Frank'],  
                'Eve': ['Grace'] }
```

- Create a **dict** with grandparents as keys, and grandchildren as values
- Desired output:

```
grandparents_dict = { 'Khaled' : ['Frank'],  
                     'Daniel' : ['Chibundu', 'Jesmyn', 'Grace'] }
```

The background is a solid orange color. It is decorated with various hand-drawn geometric shapes and lines in white and black. These include a dashed line in the top left, a white triangle in the top center, a black zigzag line in the top right, a white circle in the top right, two parallel black lines in the top right, a white triangle in the top right, a black plus sign in the bottom left, a white circle in the bottom center, a white triangle in the bottom center, a black plus sign in the bottom center, a black circle in the bottom center, and a white circle in the bottom right.

# Questions Before We Begin?

The background is a solid pink color. It is decorated with several hand-drawn elements: a dashed line in the top left, a white triangle in the top center, a black zigzag line in the top right, a white circle in the top right, two parallel black lines in the top right, a white triangle in the top right, a black plus sign in the bottom left, a white circle in the bottom center, a white triangle in the bottom center, a black plus sign in the bottom center, a black circle in the bottom center, and a white circle in the bottom right.

Fun stuff






# Find Grandchildren

- Given a **dict** with parents as keys, and lists of their children as values:  

```
parents_dict = { 'Khaled' : ['Chibundu', 'Jesmyn'],  
                'Daniel' : ['Khaled', 'Eve'],  
                'Jesmyn' : ['Frank'],  
                'Eve': ['Grace'] }
```
  - Just for kicks: what will the following print out?  

```
print(parents_dict[parents_dict['Khaled'][1]][0])
```
- 

# Find Grandchildren

- Given a **dict** with parents as keys, and lists of their children as values:

```
parents_dict = { 'Khaled' : ['Chibundu', 'Jesmyn'],  
                'Daniel' : ['Khaled', 'Eve'],  
                'Jesmyn' : ['Frank'],  
                'Eve': ['Grace'] }
```

- Just for kicks: what will the following print out?

```
print(parents_dict[parents_dict['Khaled'][1]][0])
```

*# Output: Frank*

# Find Grandchildren

+

- Given a **dict** with parents as keys, and lists of their children as values:  

```
parents_dict = { 'Khaled' : ['Chibundu', 'Jesmyn'],  
                'Daniel' : ['Khaled', 'Eve'],  
                'Jesmyn' : ['Frank'],  
                'Eve' : ['Grace'] }
```
- Just for kicks: what will the following print out?  


```
print(parents_dict[parents_dict['Khaled'][1]][0])
```
- `parents_dict['Khaled']` is `['Chibundu', 'Jesmyn']`
- `parents_dict['Khaled'][1]` is `'Jesmyn'`
- `parents_dict[parents_dict['Khaled'][1]]` is `['Frank']`
- `parents_dict[parents_dict['Khaled'][1]][0]` is `'Frank'`

The background is a solid pink color. It is decorated with various hand-drawn geometric shapes and lines in white and black. In the top left, there is a dashed white line and a solid white triangle. In the top center, there is a dashed black line and a solid black zigzag line. In the top right, there is a solid black circle and two parallel solid black lines. In the middle left, there is a solid white triangle. In the middle right, there is a solid white triangle. In the bottom left, there is a solid black plus sign. In the bottom center, there is a solid black circle and a solid white triangle. In the bottom right, there is a solid black circle and a solid white circle.

**What's Next?**



# More Python at FHDA

- De Anza's Python Sequence:
    - CIS 40, Introduction to Programming in Python
    - CIS 41A, Python Programming
    - CIS 9, Introduction to Data Science
    - CIS 41B, Advanced Python Programming
  - Foothill's Python Sequence:
    - CS 3A, Object Oriented Programming Methodologies in Python
    - CS 3B, Intermediate Software Design in Python
    - CS 3C, Advanced Data Structures and Algorithms in Python
    - CS 48A, Data Visualization
- 

The background is a solid pink color. It is decorated with various hand-drawn geometric shapes and lines in white and black. In the top left, there is a dashed white line and a solid white triangle. In the top center, there is a dashed black line and a solid black triangle. In the top right, there is a solid black zigzag line, a small white circle, and two parallel black lines. In the middle right, there is a small white triangle. In the bottom left, there is a solid black plus sign. In the bottom center, there is a small white circle and a solid white triangle. In the bottom right, there is a solid black plus sign, a solid white circle, and a solid black circle. The text "That's all, folks!" is written in a bold, white, sans-serif font in the center of the image.

**That's all, folks!**