

Intro to Coding with Python– Dictionaries

Dr. Ab Mosca (they/them)

Friday

- No class this Friday – review material

Plan for Today

- Dictionaries
 - motivation
 - defining a dictionary
 - converting multiple lists \leftrightarrow dictionaries

Recap: 15- minute exercise

Write a program that:

- asks the user to **input()** names one at a time
- adds each new name to a list called **friends**
- and after each new name is added prints the list in alphabetical order

The program should loop until the user types "**DONE**"

Motivation

- Imagine we want to use the previous exercise to create a contact list. Could do it with **multiple lists**:

```
*Untitled*
def main():

    instruction = "ADD"
    friends = []
    numbers = []

    while (instruction != "DONE"):
        # Get information about new contact
        friends.append(input("Name? "))
        numbers.append(input("Number? "))

        # Ask for next instruction
        instruction = input("ADD or DONE?")

if __name__ == "__main()__":
    main()

Ln: 9 Col: 15
```

Motivation

- If we want to access the data later:

```
*Untitled*  
print(friends[0]) # Joe  
print(numbers[0]) # 413-286-3712  
  
print(friends[1]) # Ali  
print(numbers[1]) # 972-272-2782  
  
print(friends[2]) # Clio  
print(numbers[2]) # 291-288-2897  
  
Ln: 8 Col: 32
```

- Or worse, modify it...

```
*Untitled*  
print(friends.pop(1)) # Bye,  
print(numbers.pop(1)) # Ali!  
  
Ln: 2 Col: 28
```

Motivation

- If we want to access the data later:

```
*Untitled*  
print(friends[0]) # Joe  
print(numbers[0]) # 413-286-3712  
  
print(friends[1]) # Ali  
print(numbers[1]) # 972-272-2782  
  
print(friends[2]) # Clio  
print(numbers[2]) # 291-288-2897  
  
Ln: 8 Col: 32
```

- Or worse, modify it.

```
*Untitled*  
print(friends.pop(1)) # Bye,  
print(numbers.pop(1)) # Ali!  
  
Ln: 2 Col: 28
```

What we really want

- Each name should “map” to the corresponding number:

“Joe” → “413-286-3712”

“Ali” → “972-272-2782”

“Clio” → “291-288-2897”

- That way, we could access the **number** using the **name**:

```
contacts["Joe"] # "413-286-3712"
```


Introducing: dictionaries

- **lists** were **ordered** sets of objects, and we accessed their contents via position (index)
- **dictionaries** are **unordered** sets, and we can access their contents via **keys**
- We declare them using `{...}` ← “curly braces” like this:

```
contacts = {}
```

Contacts: take 2

```
*Untitled*
def main():

    instruction = "ADD"
    contacts = {}

    while (instruction != "DONE"):
        # Get information about new contact
        new_friend = input("Name? ")
        new_number = input("Number? ")

        # Add contact to dictionary
        contacts[new_friend] = new_number

        # Ask for next instruction
        instruction = input("ADD or DONE?")

if __name__ == "__main()__":
    main()

Ln: 18 Col: 10
```

Contacts: take 2

```
*Untitled*
def main():

    instruction = "ADD"
    contacts = {}

    while (instruction != "DONE"):
        # Get information about new contact
        new_friend = input("Name? ")
        new_number = input("Number? ")

        # Add contact to dictionary
        contacts[new_friend] = new_number

        # Ask for next instruction
        instruction = input("ADD or DONE?")

if __name__ == "__main__":
    main()

Ln: 18 Col: 10
```

Contacts: take 2

```
*Untitled*
def main():

    instruction = "ADD"
    contacts = {}

    while (instruction != "DONE"):
        # Get information about new contact
        new_friend = input("Name? ")
        new_number = input("Number? ")

        # Add contact to dictionary
        contacts[new_friend] = new_number

        # Ask for next instruction
        instruction = input("ADD or DONE?")

if __name__ == "__main__()__":
    main()

Ln: 18 Col: 10
```

Contacts: take 2

```
*Untitled*
def main():

    instruction = "ADD"
    contacts = {}

    while (instruction != "DONE"):
        # Get information about new contact
        new_friend = input("Name? ")
        new_number = input("Number? ")

        # Add contact to dictionary
        contacts[new_friend] = new_number

        # Ask for next instruction
        instruction = input("ADD or DONE?")

if __name__ == "__main__()__":
    main()

Ln: 18 Col: 10
```

Interesting
dilemma

What happens when we **iterate** over a
dictionary?



```
*demo10.py - /Users/jcrouser/G...  
for c in contacts:  
    print(c)  
Ln: 1 Col: 6
```

dictionary
methods:
.keys()

- If you want to get a list of the **keys** in a **dictionary**

A screenshot of a code editor window titled '*demo10.py - /Users/jcrouser/Google Drive/Teaching/Course M...'. The editor contains two lines of Python code: 'print(contacts.keys())' and '# ["Joe", "Ali", "Clio"]'. The status bar at the bottom right of the editor shows 'Ln: 1 Col: 1'.

```
*demo10.py - /Users/jcrouser/Google Drive/Teaching/Course M...  
print(contacts.keys())  
# ["Joe", "Ali", "Clio"]  
Ln: 1 Col: 1
```

dictionary
methods:
.values()

- If you want a **list** of the **values** in a **dictionary**

```
*demo10.py - /Users/jcrouser/Google Drive/Teaching/Course M...  
print(contacts.values())  
# ["413-286-3712",  
#  "972-379-2782",  
#  "297-288-2897"]  
Ln: 4 Col: 19
```


dictionary
methods:
.items()

- If you want a **list** of the **key, value** pairs in a **dictionary**



```
*demo10.py - /Users/jcrouser/Google Drive/Teaching/Course Materi...  
for key, value in contacts.items():  
    print(key, value)  
Ln: 2 Col: 21
```

dictionary
methods:
.copy()

- If you want to **copy** the **dictionary** :

```
*demo10.py - /Users/jcrouser/Google Drive/Teaching/Course Ma...
contacts = {"Joe" : "413-286-3712",
            "Ali" : "972-379-2782",
            "Clio" : "297-288-2897"}

contacts2 = contacts.copy()
```

Ln: 5 Col: 25

just like with
a list

The `zip (...)` function

- If you want to combine two **lists** into one **dictionary**, use a comprehension and the `zip (...)` function:

```
*Untitled*
names = ["Joe", "Ali", "Clio"]

numbers = ["413-286-3712",
           "972-272-2782",
           "291-288-2897"]

contacts = {name:number for name, number in zip(names, numbers)}
```

Ln: 7 Col: 35



Recap

- **strings:** **immutable** ordered collections of characters
- **lists:** **mutable** ordered collections of objects
- **dictionaries:** **mutable** unordered collections of objects

15 minute exercise

- Write a program that ...
 - Asks the user "ADD or DONE? "
 - If the user says ADD
 - Take user input to creates a contact book entry that includes (1) name, (2) number, and (3) address
 - Ask "ADD or DONE? " again
 - If the user says DONE
 - Quit and print "X's number is Y and they live at Z" for each entry in the contact book