Dictionaries



Admin



-
- How is the project going?
- Reading assignment: please read chapter 6 with these slides
- https://medium.com/@TusharKanjariya/i-stopped-vibe-coding-for-one-day-a376f04fcf0c
-

Storing data



- Review:
  - How do we store data in our programs up till now?
  - Several possibilities – several “lucky volunteers”

Storing data



- Review:
  - How do we store data in our programs up till now?
  - Several possibilities – several “lucky volunteers”
  - Variables
  - Lists
  - Tuples
  - Anything else?
  - What is each good for?

Dictionary



- What is a dictionary in “real life”
-

Dictionary



- What is a dictionary in “real life”
  - Take a word (a key), use it to find a value:
  - What is that value?

Dictionary



- What is a dictionary in “real life”
  - Take a word (a key), use it to find a value:
  - What is that value:
    - The definition of the word
- In python we also have a dictionary type
  - Use it to associate several key=>Value pairs.
  - Use the Key to look up the value

Literal dictionary



- To type a dictionary into your code directly use {}
- To including some starting key→value pairs in the dictionary
- {<key> : <value>, ... , <keyn>: <valuen> }
  - See left of a python dictionary containing actual English dictionary entries.
  - The keys are highlighted for easier reading

- empty_dictionary= {}
-

```
#all definitions copyright by and taken from Oxford English Dictionary
word_definitions = {"world":"the earth, together with all of its countries, peoples, and natural features.",
"arc": "a part of the circumference of a circle or other curve.",
 "leg": "each of the limbs on which a person or animal walks and stands."}
```

-

Dictionary keys



- Two requirements for something to be a dictionary key
  - A key can only appear once in the dictionary
    - If you try to add it again, the value just gets changed.
  - A key must be immutable (it must be of a type that can’t change)
    - So what kind of thing (what types) can be a dictionary key that we have seen so far?

Dictionary keys



- Two requirements for something to be a dictionary key
  - A key can only appear once in the dictionary
    - If you try to add it again, the value just gets changed.
  - A key must be immutable (it must be of a type that can’t change)
    - So what kind of thing (what types) can be a dictionary key that we have seen so far?
      - Integer (4 better always be four or we will be in trouble
      - Floating point numbers
      - Strings
      - Tuples
  - Strings are by far the most common key for dictionaries.

Dictionaries to represent objects



- Use dictionaries (dicts) in python the way you use a struct in some languages
  - As a way to model a real world object as a data object in your program
- Two popular first data objects
  - Students
  - Bank accounts
  - Both can be represented with strings, floating point numbers and integers

```
my_account = {'name': "John Santore",
        'account_num': 123456,
        'balance':123.45}
```

-
-

```
the_record = {'name': "Stu Dent",

        'gpa': 3.4,
        'studentID': 11112222}
```

Lets draw them out on the board.

Changing the value in a dictionary



- Changing a value in a dictionary looks a lot like using a list,
  - Except you use the key in the [ ] instead of a number
  - See code to the right

```
my_account = {'name': "John Santore",
              'account_num': 123456,
              'balance':123.45}
```

-

```
my_account['name'] = 'Imelda Santore'
```

-

```
the_record = {'name': "Stu Dent",
              'gpa': 3.4,
              'studentID': 11112222}
```

- #maybe Stu's grades went up

```
the_record['gpa'] = 3.6
```

Accessing/checking value of dictionary



- Getting a value from a dictionary works the same way
  - use the key inside [ ] to retrieve the value

```
the_record = {'name': "Stu Dent",
              'gpa': 3.4,
              'studentID': 11112222}

print(f"our student is named {the_record['name']}")

current_gpa = the_record['gpa']
```

Adding a new key-value pair



- Often we need to add a new key value pair
  - For example you really need a number of credits for any student record.
  - So lets add it

```
the_record = {'name': "Stu Dent",
              'gpa': 3.4,
              'studentID': 11112222}
```

Adding a new key-value pair



- Often we need to add a new key value pair
  - For example you really need a number of credits for any student record.
  - So lets add it
  - Do the same as in the previous approaches
    - dictionary[<new key>] = new value
  -

```
the_record = {'name': "Stu Dent",
              'gpa': 3.4,
              'studentID': 11112222}
```

```
the_record['credits'] = 135
```

Dictionaries that represent a collection of values



- While we can use a single dictionary to represent an object
  - Often we want to use a dictionary to collect a bunch of related key-value pairs
- For example this list of games associated with the thousands of players in early Aug 2025
- (eg: Stardew Valley had 81,000 players.)

```
top_games={"counter strike2": 1100,
        "DOTA2": 571,
        "PUBG:Battlegrounds":334,
        "Rust": 123,
        "Delta Force": 106,
        "Banana": 104,
        "Stardew Valley": 81}
```

Looping through the dictionary



- Sometimes we want to examine every item in the dictionary
  - Instead of using just a value here or there
- Can loop through dictionary keys like looping through a list
  - for <new variable> in dictionary.keys()
- Let's look at what is going on over here
- Maybe 'hand execute' on board for he first few

```
top_games={"counter strike2": 1100,
        "DOTA2": 571,
        "PUBG:Battlegrounds":334,
        "Rust": 123,
        "Delta Force": 106,
        "Banana": 104,
        "Stardew Valley": 81}
```

```
for game in top_games.keys():
    print(f"Game: {game} had
{top_games[game]*1000} daily players in August")
```

In class exercise



- Take the top_games dictionary and loop through the keys to find the game with the most players
  - We can see because we wrote it out, but if the dictionary gets updated, the answer might change
  - Then print out the name of the game with the most players.
  - Then lets look up the games for this week and add in a couple more games and run our program again

- top_games={"counter strike2": 1100,
  "DOTA2": 571,
  "PUBG:Battlegrounds":334,
  "Rust": 123,
  "Delta Force": 106,
  "Banana": 104,
  "Stardew Valley": 81}

Dictionary keys that don’t exist



- Suppose we try to use a key that doesn’t exist
-
- We might try the code to the right if let the AI do too much suggesting
  - And the highlighted code will cause trouble
  -

```
top_games={"counter strike2": 1100,
        "DOTA2": 571,
        "PUBG:Battlegrounds":334,
        "Rust": 123,
        "Delta Force": 106,
        "Banana": 104,
        "Stardew Valley": 81}
```

```
counter_strike2_players = top_games["counter strike2"]
next_players = top_games["counter strike3"]
if counter_strike2_players < next_players:
    print("We've moved on")
```

Key Error



- When you try to directly access a key that doesn’t exist in the dictionary, you get an error like this
- Traceback (most recent call last):
- File "/workspace/Fall2025/CS1/dearPyGuiTesting/dictionaryDemo.py", line 13, in <module>
- next_players = top_games["counter strike3"]
- ~~~~~~~~~~^^^^^^^^^^^^^^^^^^
- KeyError: 'counter strike3'
-

Using get() to avoid the error



- Like lists, dictionaries are also objects, so there are functions we can use to ask the dictionary to do something for us.
  - Like the keys function from a couple of slides ago.
  - So what is the value of counter_strike2_players?

```
top_games={"counter strike2": 1100,
        "DOTA2": 571,
        "PUBG:Battlegrounds":334,
        "Rust": 123,
        "Delta Force": 106,
        "Banana": 104,
        "Stardew Valley": 81}
```

```
counter_strike2_players = top_games.get("counter strike2")
next_players = top_games.get("counter strike3")
```

Using get() to avoid the error



- Like lists, dictionaries are also objects, so there are functions we can use to ask the dictionary to do something for us.
  - Like the keys function from a couple of slides ago.
  - So what is the value of counter_strike2_players?
    - 1100 right?
  - So what about next_players?
    - We haven't talked about it, but any guesses?

```
top_games={"counter strike2": 1100,
        "DOTA2": 571,
        "PUBG:Battlegrounds":334,
        "Rust": 123,
        "Delta Force": 106,
        "Banana": 104,
        "Stardew Valley": 81}
```

```
counter_strike2_players = top_games.get("counter strike2")
next_players = top_games.get("counter strike3")
```

Using get() to avoid the error



- Like lists, dictionaries are also objects, so there are functions we can use to ask the dictionary to do something for us.
  - Like the keys function from a couple of slides ago.
  - So what is the value of counter_strike2_players?
    - 1100 right?
  - So what about next_players?
    - None
    - Which is python speak for no object

```
top_games={"counter strike2": 1100,
        "DOTA2": 571,
        "PUBG:Battlegrounds":334,
        "Rust": 123,
        "Delta Force": 106,
        "Banana": 104,
        "Stardew Valley": 81}
```

```
counter_strike2_players =
top_games.get("counter strike2")
next_players = top_games.get("counter
strike3")
```

None



- None is python's way of saying no object here
- By default python returns None from functions that return an object when there is no object to return.
- None will evaluate to False in an if statement
  - See code

```
top_games={"counter strike2": 1100,
        "DOTA2": 571,
        "PUBG:Battlegrounds":334,
        "Rust": 123,
        "Delta Force": 106,
        "Banana": 104,
        "Stardew Valley": 81}

for game in top_games.keys():
    print(f"Game: {game} had
{top_games[game]*1000} daily players in August")

if top_games.get("counter strike3"):
    print(f"counter strike3 has
{top_games.get("counter strike3")} players")
else:
    print(f"no such game")
```

Get with a default value



- If you don’t want None when an item doesn’t exist in the dictionary, then get can take a default value
  - What would be a good default value for players in our games/players dictionary?

Get with a default value



- If you don’t want None when an item doesn’t exist in the dictionary, then get can take a default value
  - What would be a good default value for players in our games/players dictionary?
  - I think 0 makes the most sense, since if the game isn’t there, they likely have no players.
  - Put the default value as the second thing in the parentheses

```
top_games={"counter strike2": 1100,
        "DOTA2": 571,
        "PUBG:Battlegrounds":334,
        "Rust": 123,
        "Delta Force": 106,
        "Banana": 104,
        "Stardew Valley": 81}

for game in top_games.keys():
    print(f"Game: {game} had
{top_games[game]*1000} daily players in
August")

cs3_players = top_games.get("counter strike3",0)

#now cs3_players is 0 instead of None
```

Looping through dictionary to print



- If you want to loop through and print the dictionary
  - Then you need both keys and values
  - Use the items function in the dictionary
    - With a for loop
  - This will create two variables
    - First for key, then for matching value
    - See example

```
top_games={"counter strike2": 1100,
        "DOTA2": 571,
        "PUBG:Battlegrounds":334,
        "Rust": 123,
        "Delta Force": 106,
        "Banana": 104,
        "Stardew Valley": 81}

for game, players in top_games.items():
    print(f"{game} had  {players} in August")
```

Lets put it all together



- Lets build a small program together that puts all the dictionary work together with our loops, if statements, files and more.
- Grab two files
  - WordFile.txt from the resources page
  - https://www.gutenberg.org/cache/epub/2600/pg2600.txt
  - Save them both to your current pycharm project
  - Then create a new python file
- Project description on next line once everyone has these two files downloaded.
  - Let me and/or the PALs know if there is trouble

Word count exercise



- Next we will open a file (lets start with WordFile.txt)
  - And read in all of the lines
  - Break each line up into words
  - Strip the newlines off any words that have them
  - Remove punctuation from the words

```
import string
translator = str.maketrans('', '', string.punctuation)
# Remove punctuation
clean_text = s.translate(translator)
```

  - If the word is in the word_count dictionary, then update the count
  - Otherwise add it to the word_count
  - Finally when everything is done, report the word counts to the user by printing them all out.

Dictionaries in a list



- Dictionaries are just more data
- So if we had a bunch of dictionaries each representing a student, we could build a list which represented all the students in a class
- Show results on board

```
the_record = {'name': "Stu Dent",
              'gpa': 3.4,
              'studentID': 11112222}

student1 = {'name' : "Hard Worker",
            'gpa': 3.8,
            'studentID': 12121212}

student2 = {'name' : "Playstoo Muchgames",
            'gpa': 2.01,
            'studentID': 121123123}

all_students = []
all_students.append(student1)
all_students.append(student2)
all_students.append(the_record)
```

Reminder:Reading



- Read chapter 6
- We covered most of it here
  - But there are a couple of things in chapter 6 in your book