

# Python Lesson 5: Dictionaries and JSON

[vanderbi.it/py](http://vanderbi.it/py)

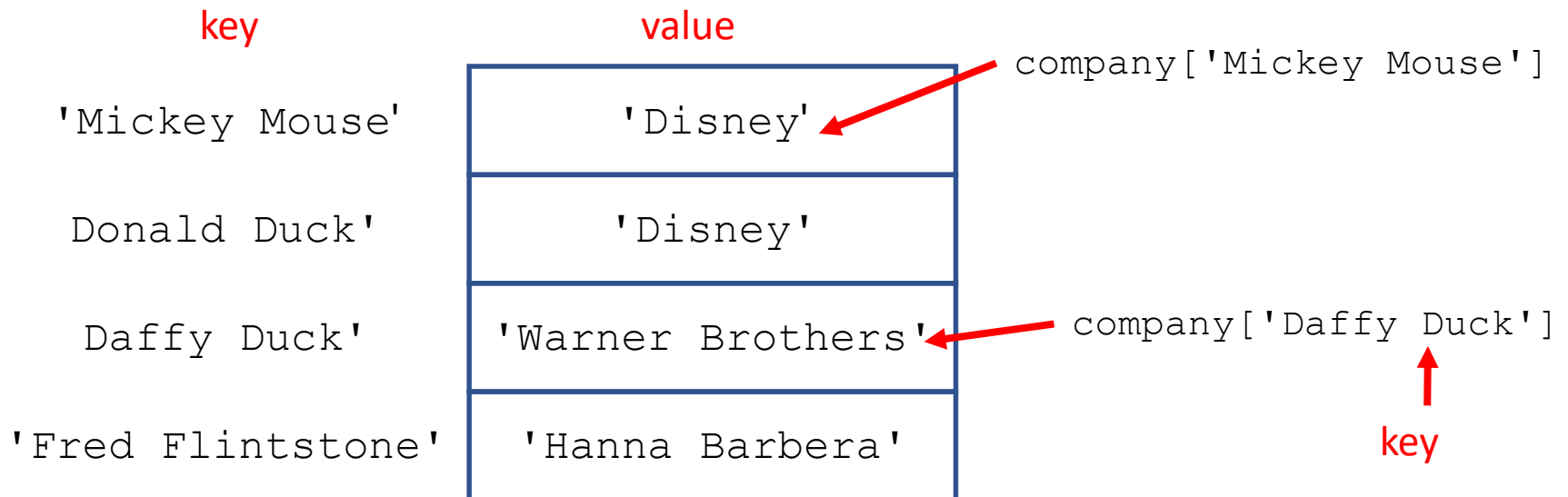
Steve Baskauf



# Dictionaries

- Dictionaries are an unordered data structure.
- They're defined using curly brackets: { }
- Values are identified by keys.
- We "look up" values in the dictionary using the keys.

```
company = {'Mickey Mouse':'Disney', 'Donald Duck':'Disney', 'Daffy Duck':'Warner Brothers', 'Fred Flintstone':'Hanna Barbera'}
```



# try...except... for error trapping

```
try:  
    code that might throw an error goes here  
except:  
    code to be executed if there's an error goes here  
here's where the code execution continues
```

- Error trapping handles problems gracefully instead of having the script crash.
- An error is called an **exception**.
- Code blocks are identified by indentation (as usual)
- Colons required after **try** and **except**

# try...except... for error trapping

- Example:

```
try:
    print('That character works for ' + company[characterName])
except:
    print("I don't know who that character works for.")
print("That's all folks!")
```

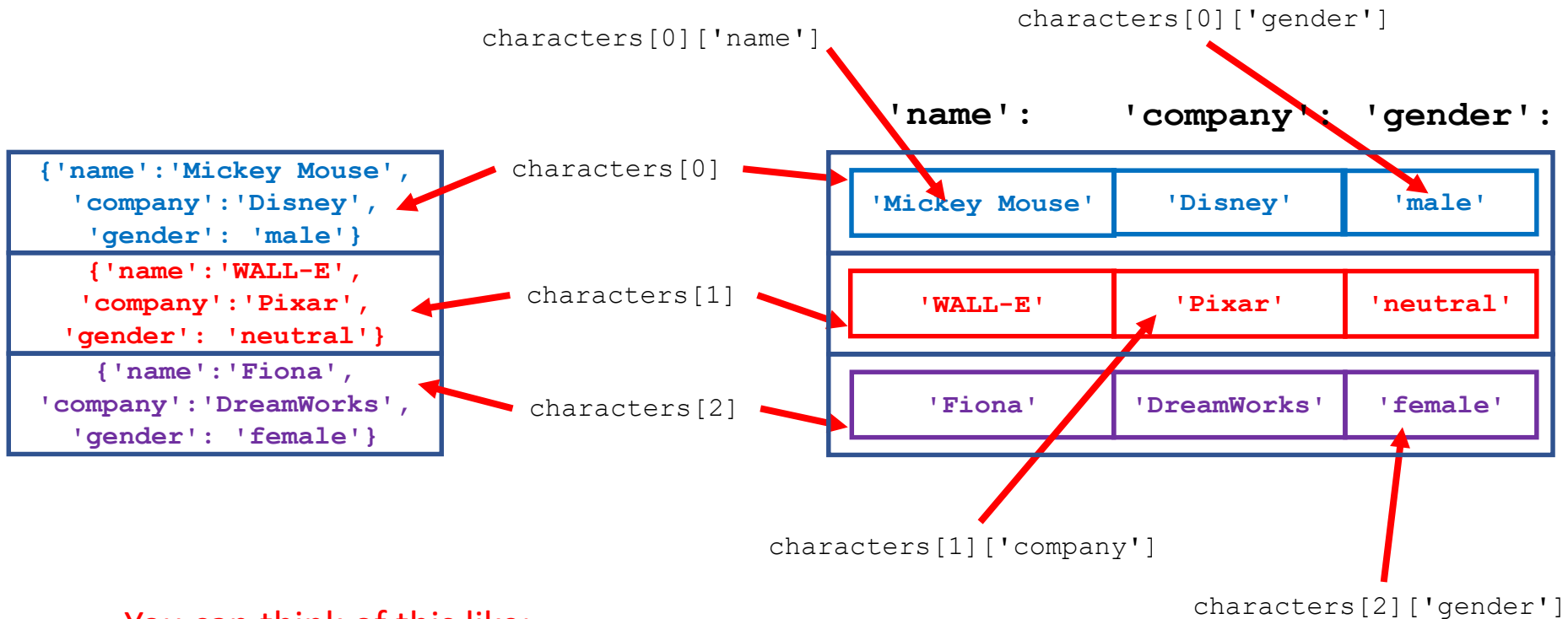
- It's a good idea to error trap any error that can be predicted to happen sometime (e.g. file not found)

# Try this

- Note: the keys in a dictionary should be unique
- It is typical for keys to either:
  - be some kind of identifier for a thing (this example)
  - be some kind of characteristic of the thing (next example)

# Lists of dictionaries

```
characters = [{'name': 'Mickey Mouse', 'company': 'Disney', 'gender': 'male'}, {'name': 'WALL-E', 'company': 'Pixar', 'gender': 'neutral'}, {'name': 'Fiona', 'company': 'DreamWorks', 'gender': 'female'}]
```



You can think of this like:

**data [row] [key]**

Since the keys aren't ordered, there is no significance to the location of the columns.

# Lists of dictionaries (cont.)

- Lists are iterable. Dictionaries aren't (they are unordered).
- It's common for each item on the list to represent an individual of some category of thing and each key:value pair in that individual's dictionary to represent a property of that individual.
- Stepping through the list processes each individual.

# Examples



# What is JSON?

- A basic unit of JSON is a **key:value pair**. For example:

`"name": "Steve"` (strings must be in quotes)

`"fingers": 10` (numbers don't need quotes)

- A **JSON object** is a list of key:value pairs inside curly brackets.

```
{ "name": "Steve", "fingers": 10,  
  "street": "Penny Lane" }
```

- Multiple values can be put in an **array** inside square brackets.

```
[ "Steve", "Steven", "Esteban" ]
```

# Nesting in JSON

- Arrays can be nested inside objects

```
{  
  "name":  
    [  
      "Steve",  
      "Steven",  
      "Esteban"  
    ],  
  "fingers":10,  
  "street":"Penny Lane"  
}
```

- We use this when there are **multiple options** for a **value**
- In this example, the array holds multiple name values.

# Whitespace

- Whitespace is not important – it can be used to make the JSON structure clearer. The following mean exactly the same thing:

```
{"name":["Steve","Steven","Esteban"], "fingers":10, "street":"Penny Lane"}
```

```
{"name":["Steve","Steven","Esteban"],  
 "fingers":10,  
 "street":"Penny Lane"}
```

```
{  
  "name":  
    [  
      "Steve",  
      "Steven",  
      "Esteban"  
    ],  
  "fingers":10,  
  "street":"Penny Lane"  
}
```

# Nesting in JSON

- Objects can be nested inside arrays

```
[  
  {  
    "created_at": "Wed Sep 18 19:50:41 +0000 2019",  
    "text": "The \u201cdigital downloads\u201d tax makes an appearance!",  
    "lang": "en"  
  },  
  {  
    "created_at": "Wed Sep 18 19:28:44 +0000 2019",  
    "text": "I couldn't feel my fingertips this morning it was so cold!",  
    "lang": "en"  
  },  
  {  
    "created_at": "Wed Sep 18 14:08:54 +0000 2019",  
    "text": "RT @wnprwheelhouse: @wnprharriet giving shoutout to @wnpr !",  
    "lang": "en"  
  }  
]
```

- Use this to assign properties and values to **multiple items**
- In this example, each item is a described tweet

# Nesting in JSON

- Objects can be nested inside objects

```
{
  "in_reply_to_screen_name": null,
  "user":
    {
      "id": 6253282,
      "id_str": "6253282",
      "name": "Carmen Baskauf",
      "screen_name": "cbaskauf",
      "location": "Hartford, CT"
    }
  ,
  "geo": null,
  "coordinates": null
}
```

- We use this when a value needs to be **further described** using additional properties.
- In this example, the inner object describes the user

# JSON converted to Python objects

- The `json.loads()` function turns a JSON string into a Python data object.
- Example: lists nested inside dictionaries

```
data = json.loads('''
{
    "name":
        [
            "Steve",
            "Steven",
            "Eteban"
        ],
    "fingers":10,
    "street":"Penny Lane"
}
''')
```

```
>>> print( data['name'][1] )
```

Steven

# JSON converted to Python objects

- Example: dictionaries nested inside arrays

```
data = json.loads('''
[
  {
    "created_at": "Wed Sep 18 19:50:41 +0000 2019",
    "text": "The \u201cdigital downloads\u201d tax makes an appearance!",
    "lang": "en"
  },
  {
    "created_at": "Wed Sep 18 19:28:44 +0000 2019",
    "text": "¡No podía sentir las yemas de mis dedos esta mañana, hacía tanto
frío",
    "lang": "es"
  },
  {
    "created_at": "Wed Sep 18 14:08:54 +0000 2019",
    "text": "RT @wnprwheelhouse: @wnprharriet кричать @wnpr !",
    "lang": "ru"
  }
]
''')

>>> print( data[1][ 'lang' ] )

es
```

# JSON converted to Python objects

- Example: dictionaries nested inside dictionaries

```
data = json.loads('''
{
    "in_reply_to_screen_name": null,
    "user":
        {
            "id": 6253282,
            "id_str": "6253282",
            "name": "Carmen Baskauf",
            "screen_name": "cbaskauf",
            "location": "Hartford, CT"
        }
    ,
    "geo": null,
    "coordinates": null
}
''')
```

```
>>> print( data['user']['location'] )
Hartford, CT
```



# General pattern

- In complex JSON structures, inner structures can be nested inside outer structures – potentially many times.
- In a variable, we describe the path from outer to inner structures through a series of square brackets.
- If the next structure is a JSON array (Python list), we use an index number.
- If the next structure is an JSON object (Python dictionary), we use a key string.

Try this