Object Processing

Guttag Chapter 10

Goals

1. Recall terms and concepts related to classes

2. Look at another example

- Handle raw data
- Define a class that is a matchmaking service for professionals!
  - search and match people by profession
  - search and match people by country of residence
- Query and match professionals with their attributes

3. Explore code

But first, More on Computer and Information Science Majors

Goals

1. Recall terms and concepts related to classes

2. Look at another example

- Handle raw data
- Define a class that is a matchmaking service for professionals!
  - search and match people by profession
  - search and match people by country of residence
- Query and match professionals with their attributes

3. Explore code

1. Recall terms and concepts related to classes

1. Recall Terms

- Class
- Constructor
- Self
- Attributes
- Methods
- Dunder Method
- Public and private naming convention
- Instantiation
- Dot notation to access methods and attributes

```
class Vehicle():
  """Abstract data type representing a vehicle."""

  def __init__(self, num_seats: int, num_doors: int, engine_type: str):
    """Define the constructor."""
    self._seats = num_seats
    self._doors = num_doors
    self._engine = engine_type
    self._milage = 0.0

  def drive(self, num_miles: float):
    """Add milage to the vehicle."""
    self._milage += num_miles
    return None

  def milage(self):
    """Get the milage of the vehicle."""
    return self._milage

  def __repr__(self):
    """Define the printable representation of the vehicle."""
    return f"{self._engine} vehicle with {self._seats} seats, " +\
           f"{self._doors} doors, and {self._milage} miles."
```

1. Recall Concepts

- The data and the methods that manage and operate on the data are together!

```
class Vehicle():
  """Abstract data type representing a vehicle."""

  def __init__(self, num_seats: int, num_doors: int, engine_type: str):
    """Define the constructor."""
    self._seats = num_seats
    self._doors = num_doors
    self._engine = engine_type
    self._milage = 0.0

  def drive(self, num_miles: float):
    """Add milage to the vehicle."""
    self._milage += num_miles
    return None

  def milage(self):
    """Get the milage of the vehicle."""
    return self._milage

  def __repr__(self):
    """Define the printable representation of the vehicle."""
    return f"{self._engine} vehicle with {self._seats} seats, " +\
           f"{self._doors} doors, and {self._milage} miles."
```

2. Another Example

Class to support professionals

Task

- store data in a convenient way
- display data in a convenient way
- add functionality HR (Human Resources) to match data

Data

- the raw data is like this:

```
Samantha Rhodes,Maldives,(912)136-3882,"Research officer, trade union",terryjames@example.net
Matthew Johnson,Iran,001-366-114-0721x393,Agricultural consultant,molly98@example.com
Summer Stewart,Jamaica,2494404249,Economist,ovazquez@example.com
William Anderson,United States Virgin Islands,+1-888-635-0096x9565,Museum/gallery exhibitions officer,randyhartman@example.net
Jeremy Bates,Central African Republic,+1-449-207-9863x997,Chartered public finance accountant,livingstonamanda@example.net
Jorge Wright,Peru,(970)111-7796,"Surveyor, building control",harringtonmichael@example.com
Dustin Jackson,Gambia,755-090-9702x49724,Immunologist,ijennings@example.org
Joan Paul,San Marino,(218)682-4690x416,"Surveyor, planning and development",mcdonaldrenee@example.org
```

Note on Raw Data

The data are highly structured!!!!

- Name, Country of Residence, Cell Number, Profession, Email
- each new line is a different person
- every person has the five attributes (^^^)
- every item is separated using a comma

```
Samantha Rhodes,Maldives,(912)136-3882,"Research officer, trade union",terryjames@example.net
Matthew Johnson,Iran,001-366-114-0721x393,Agricultural consultant,molly98@example.com
Summer Stewart,Jamaica,2494404249,Economist,ovazquez@example.com
William Anderson,United States Virgin Islands,+1-888-635-0096x9565,Museum/gallery exhibitions officer,randyhartman@example.net
Jeremy Bates,Central African Republic,+1-449-207-9863x997,Chartered public finance accountant,livingstonamanda@example.net
Jorge Wright,Peru,(970)111-7796,"Surveyor, building control",harringtonmichael@example.com
Dustin Jackson,Gambia,755-090-9702x49724,Immunologist,ijennings@example.org
Joan Paul,San Marino,(218)682-4690x416,"Surveyor, planning and development",mcdonaldrenee@example.org
```

Note on Raw Data

File I/O

- Getting raw data into the computer is the first step
  - Read the documentation on csvread
  - https://docs.python.org/3/library/csv.html
- Look at previous example in Integer Squaring

Start Designing a Class

- Class: Person
- Constructor: _ _init_ _(Self, Name, Country of Residence, Cell Number, Profession, Email)
- Self: conventional first parameter, referring to the object it**self**
- Attributes:

```
class Person:
    """Define a Person class."""

    def __init__(self, name: str, residence: str, cell: str, profession: str, email: str) -> None:
        """Define the constructor for a person."""
        self.name = name
        self.residence = residence
        self.cell = cell
        self.profession = profession
        self.email = email
```

Critical thinking

- are the attributes public or private?
- is the _ _init_ _ function public or private?

```
class Person:
    """Define a Person class."""

    def __init__(self, name: str, residence: str, cell: str, profession: str, email: str) -> None:
        """Define the constructor for a person."""
        self.name = name
        self.residence = residence
        self.cell = cell
        self.profession = profession
        self.email = email
```

Display the data in a convenient way

```
    def __repr__(self) -> str:
        """Define human-readable representation of the person."""
        return f"{self.name} is a {self.profession} who lives in {self.residence}. You can call this
person at {self.cell} and email them at {self.email}"

    def create_list(self) -> List[str]:
        """Create a list of strings representing the person."""
        details = []
        details.append(self.name)
        details.append(self.residence)
        details.append(self.cell)
        details.append(self.profession)
        details.append(self.email)
        return details
```

Complete Class

```python
class Person:
    """Define a Person class."""

    def __init__(self, name: str, residence: str, cell: str, profession: str, email: str) -> None:
        """Define the constructor for a person."""
        self.name = name
        self.residence = residence
        self.cell = cell
        self.profession = profession
        self.email = email

    def __repr__(self) -> str:
        """Define human-readable representation of the person."""
        return f"{self.name} is a {self.profession} who lives in {self.residence}. You can call this

    def create_list(self) -> List[str]:
        """Create a list of strings representing the person."""
        details = []
        details.append(self.name)
        details.append(self.residence)
        details.append(self.cell)
        details.append(self.profession)
        details.append(self.email)
        return details
```

Add functionality to match!

```
# define other useful functions

def find_matching_people(attribute: str, search_term: str, list_of_persons: List[Person]) -> List[Person]:
    """Collect all matching persons into a new list."""
    list_of_matching_persons = []
    for current_person in list_of_persons:
      if search_term in getattr(current_person, attribute): # https://docs.python.org/3/library/functions.html#getattr
            list_of_matching_persons.append(current_person)
    return list_of_matching_persons
```

Does this belong inside or outside the class?

Once matches are found, then what?

- display the list of matching persons
  - do slight modification on ^^ for prettiness and human readability

```
def create_display_text(list_of_persons: List[Person]) -> str:
    """Convert list of persons into a string with pretty formatting."""
    display_text = ""
    for current_person in list_of_persons:
      display_text += "- " + str(current_person) + "\n"
    return display_text
```

3. Explore Code

Summary

Methods are inside classes

Not all functions need to go inside the class

Classes can facilitate data organization and processing!