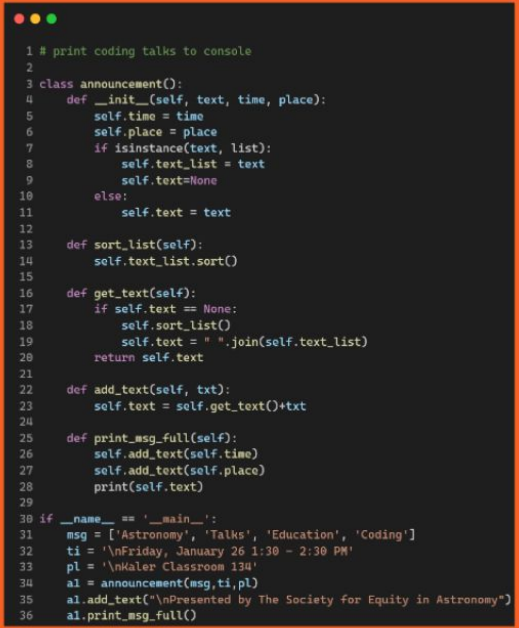
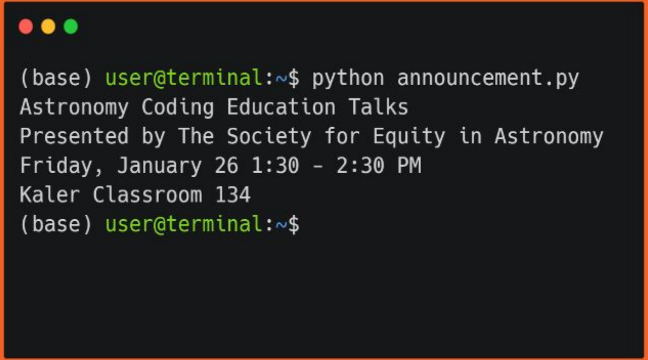
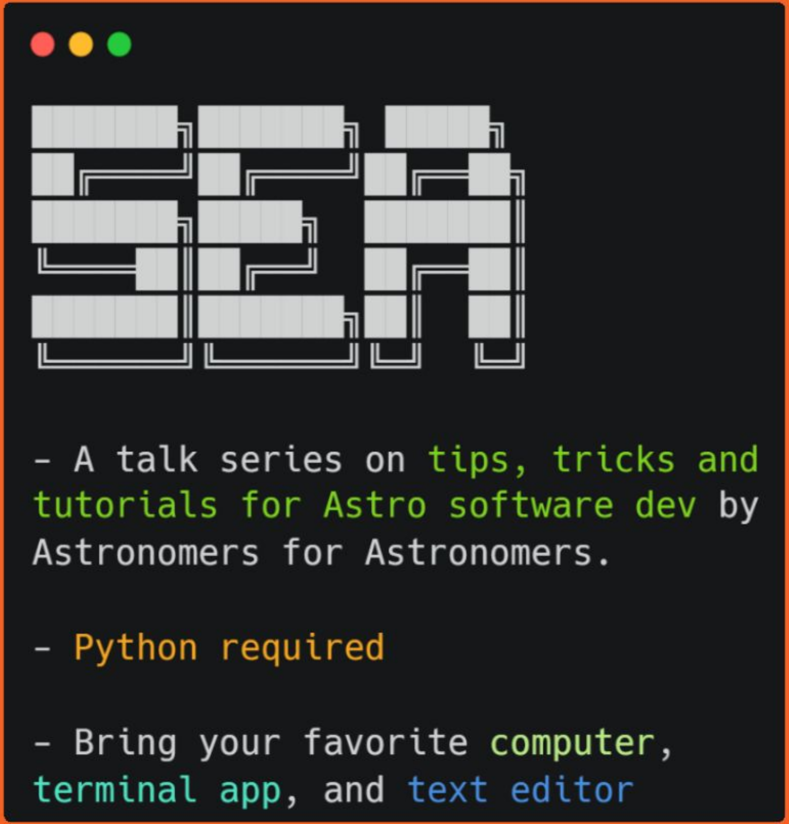


```python
# print coding talks to console

class announcement():
    def __init__(self, text, time, place):
        self.time = time
        self.place = place
        if isinstance(text, list):
            self.text_list = text
            self.text=None
        else:
            self.text = text

    def sort_list(self):
        self.text_list.sort()

    def get_text(self):
        if self.text == None:
            self.sort_list()
            self.text = " ".join(self.text_list)
        return self.text

    def add_text(self, txt):
        self.text = self.get_text()+txt

    def print_msg_full(self):
        self.add_text(self.time)
        self.add_text(self.place)
        print(self.text)

if __name__ == '__main__':
    msg = ['Astronomy', 'Talks', 'Education', 'Coding']
    ti = '\nFriday, January 26 1:30 - 2:30 PM'
    pl = '\nKaler Classroom 134'
    a1 = announcement(msg,ti,pl)
    a1.add_text("\nPresented by The Society for Equity in Astronomy")
    a1.print_msg_full()
```



```
(base) user@terminal:~$ python announcement.py
Astronomy Coding Education Talks
Presented by The Society for Equity in Astronomy
Friday, January 26 1:30 - 2:30 PM
Kaler Classroom 134
(base) user@terminal:~$
```



SEA

- A talk series on tips, tricks and tutorials for Astro software dev by Astronomers for Astronomers.

- Python required

- Bring your favorite computer, terminal app, and text editor

Object Oriented Programming

Astronomy Coding Education Talks

Presented by The Society for Equity in Astronomy

January 26, 2024

What is an Object?

- Is describable (Person, Place, or Thing)
- Features or attributes
- Inherent to the object, can differentiate it
- How would you describe this mug?
    - Texture
    - Color
    - Shape
    - Etc…

“Mug” Description

- Texture
  - Hard, rigid
  - Smooth ceramic
- Color
  - Blue
  - White text on it
- Shape
  - Concave
  - Can hold things inside it
  - Height and Diameter of rim
  - Volume determined by dimensions
- Many more…

Same can be applied to Object Oriented Programming!

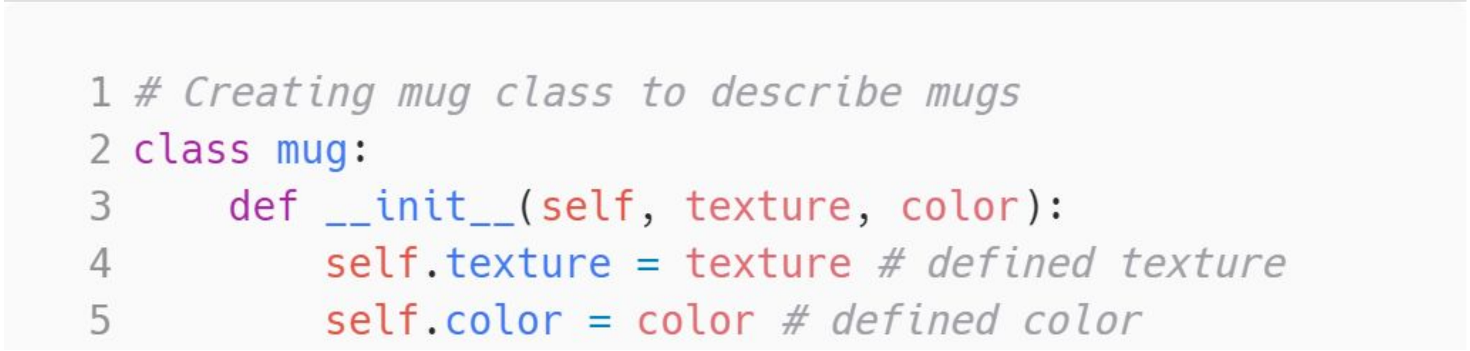
```
>>> Mug.texture
‘hard’
>>> Mug.color
‘blue’
>>> Mug.contents
‘Tea’
>>> Mug.volume_ml
350
>>> Mug.text
‘Illinois Department of Astronomy’
```

Objects with Python `class`, `__init__`, and `self`

- `class`
  - Classes in Python tie the object's constructor, attributes, and methods
  - Refer to it when initializing a new object
- `__init__`
  - This function/method is the constructor, it is the function called when you initialize a new object
  - Can have arguments that pass information to fill in attributes
  - Can define object attributes immediately when it is created
- `self`
  - `self` is the internal object that can be referred to any function within the class as long as it has the self variable as its first argument
  - Don't worry about including this in any method calls, it knows what object you're referencing!

Defining attributes using `__init__` and `self`

- Example Object Setup,



```python
# Creating mug class to describe mugs
class mug:
    def __init__(self, texture, color):
        self.texture = texture # defined texture
        self.color = color # defined color
```

- When used,

```
>>> mymug = mug(‘hard’, ‘blue’)
>>> mymug.color
‘blue’
```

Defining attributes after the `__init__`

- Python is kind enough to not need *getter* and *setter* methods to modify object attributes
- From earlier example,

```
>>> mymug = mug(‘hard’, ‘blue’)
>>> mymug.color
‘blue’
```

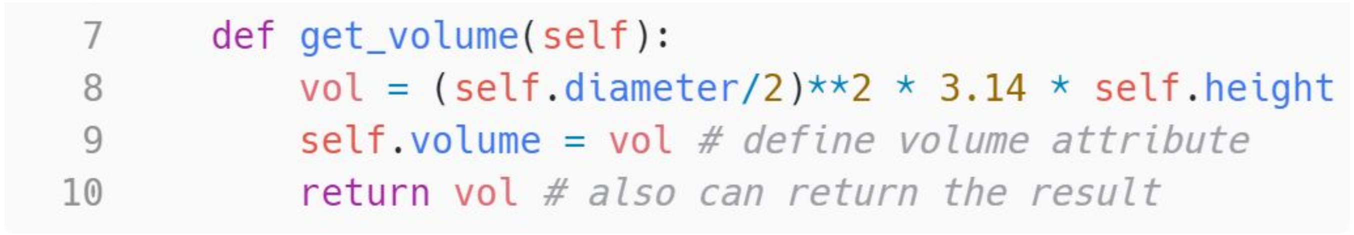
    - **`mug`** was the class name, was defined in the code
    - Used `mug()` to define (*set*) texture and color attributes immediately
    - `mymug` was individual objects variable name which we initialized
    - `mymug.color` *gets* the `color` attribute set during initialization

- To *set* new attributes or change current ones,

```
>>> mymug.height = 10.5 # in centimeters
>>> mymug.height
10.5
```

Object Methods

- Functions inside of classes
- Can use `self` to use defined attributes inside the method as variables
- Let's define the mugs diameter first as 9 cm
- Calculate the volume as a method,



```python
    def get_volume(self):
        vol = (self.diameter/2)**2 * 3.14 * self.height
        self.volume = vol # define volume attribute
        return vol # also can return the result
```

- In use,

```
>>> mymug.get_volume()
667.64
>>> mymug.volume
667.64
```

Benefits of OOP

- Intuitive for describing complex systems
- Define your own structure, not limited to pre defined structures with rules like lists, arrays, dictionaries
- Have multiple functions keep track of pre-defined variables
- User friendly

Final Tips and Tricks

- Use `objname.__dir__()` to list all object attributes and methods
    - Lots of built in ones too!
- Can call methods inside of `__init__()`
    - Also, methods are location independent, don't need to place them before or after each other in order for things to run
- Superclasses and Inheritance
    - Have other classes inherit attributes from previous classes
- Private and Hidden Methods
    - Can use "_" (single underscore) in front of a method name to hide it from method listing
    - Also can use "__" (double underscore) in front of a method name so it can't be called by the user, this is called a *Private* method, all others are *Public* methods.

Now go and use OOP in your work!

Also, Questions!