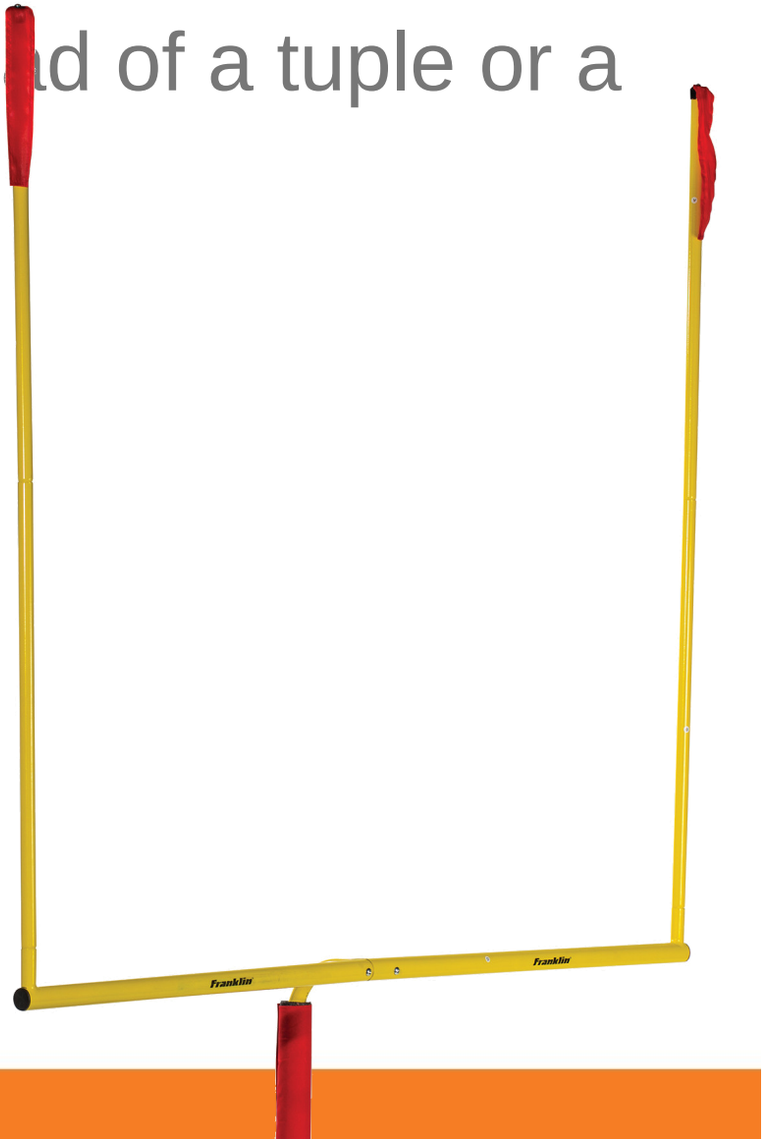
Objects

Develop Intelligence

Goals

1. Name an advantage of using an object instead of a tuple or a dictionary
2. Explain the idea of 'dunder methods'
3. Define the Python dataclass construct



Roadmap

1. What and Why?
2. Good Old OO
3. Instances
4. Encapsulation
5. Properties
6. New Hotness: the `dataclass`
7. Dunder

Develop
Intelligence

What and Why?

Develop Intelligence

Complaints about Tuples

- Immutable
- No naming for clarity
- No protection of invariants

```
1 red = (255,0,'chicken')
2 yellow = (255,275,0)
3 purple = (128,0,128)
```

Dictionary Complaints

- Only data-- no behavior
- No help from intellisense
- Awkward syntax
- No protection of invariants

```
1 yellow = {'r':755,'g':255,'b':0}
2 purple = {'r':128,'g':0,'b':'spam'}
```

Develop
Intelligence

Good Old OO

Develop Intelligence

What's an Object?

Develop Intelligence

- Associates names with
  - Data
  - Behavior
- Usual OO stuff supported (mostly):
  - Inheritance
  - Polymorphism
  - Methods
  - Properties
  - Encapsulation

Example: Color Class

- Ignore `self` for now

```
class Color:
  def __init__(self,r,g,b):
    self.red = r
    self.green = g
    self.blue = b

paint = Color(128,128,0)
print(f'Channel red: {paint.red}')
```

Better Color Class

- Protect those invariants

```
class Color:
  def __init__(self,r,g,b):
    if r < 0 or r > 255:
      raise ValueError("Something's wrong here")
    # Etc ...

yellow = Color(1128,128,0) #Error!
purple = Color(128,0,128)
```

Adding Behavior

```
class Color:
  def __init__(self,red,blue,green):
      self.red=red
      self.blue=blue
      self.green=green
  def to_hex(self):
      return f'#{self.red:x}{self.green:x}{self.blue:x}'

purple = Color(128,128,0)
print(purple.to_hex())
```

More Behavior

```
1  class Car:
2    def __init__(self,make,model):
3        self.make=make
4        self.model=model
5
6    def drive(self):
7        return f'This {self.make} is one sweet ride!'
8
9  ride = Car('volvo','240')
10 ride.drive()
```

Develop
Intelligence

Instances

Develop
Intelligence

About `self`

- Current instance is the first parameter on instance methods
- Explicit equivalent to `this` in other languages
- Called `self` by convention
- Passed by the Python runtime, **not** the caller

Static Methods

- Anything that doesn't touch an instance member

```
class Car:
    def get_makes():
      return ["volvo","chevy"]

print(Car.get_makes())

ride = Car()
print(ride.get_makes()) #Bork!
```

@staticmethod

- Makes a 'static' property accessible via instance reference

```
1 class Car:
2     @staticmethod
3     def get_makes():
4       return ['volo','chevy']
5
6 print(Car.get_makes()) #No problem!
7
8 ride = Car()
9 print(ride.get_makes()) #Ok!
```

Develop
Intelligence

Encapsulation

Develop Intelligence

Access Modifiers

- Prefix *private* members with double-underscores __
  - Enforced via 'name mangling'
- Prefix *non-public* members with underscore _
  - Convention only, no runtime enforcement

Name Mangling

```
1  class Car:
2    def __diagnose(self):
3      pass
4    def drive(self):
5      pass
6
7  car = Car()
8  for attr in dir(car):
9    if not attr.endswith('__'):
10     print(attr)
```

Develop
Intelligence

Properties

Property Getters

```
1  class Car:
2
3    def __init__(self,make,model):
4      self._make=make
5      self._model=model
6
7    @property
8    def make(self):
9      return self._make
```

Property Setters

```
1  class Person:
2    def __init__(self,age):
3      self._age = age
4
5    @property
6    def age(self):
7      return self._age
8
9    @age.setter
10   def age(self, value):
11     if value<0:
12       raise ValueError('age must be > 0')
13     self._age = value
```

Develop
Intelligence

New Hotness: the `dataclass`

Develop Intelligence

Using dataclasses

- Creates `__init__` for you
- Optionally immutable
- May contain methods like any other class
- Requires type annotation

Example

```
from dataclasses import dataclass

@dataclass(frozen=True)
class Color:
  r:int
  g:int
  b:int

  def to_hex(self):
      return f'#{self.r:x}{self.g:x}{self.b:x}'

purple = Color(128,128,0)
print(purple.to_hex())
```

Develop
Intelligence

Dunder

Develop Intelligence

Dunder Methods

- Called for the double-underscore
- Used for python internal behavior
- ***Lots*** of them

Example: DoNothing

```
1 class DoNothing:
2     pass
3
4 instance = DoNothing()
5 print(len(instance)) #Error!
```

Example: DoVeryLittle


```
1 class DoVeryLittle:
2     def __len__(self):
3         return 5
4
5 instance = DoVeryLittle()
6 print(len(instance))
```

More Dunders

Develop Intelligence

Method	What it does
`__init__`	Constructor
`__del__`	Finalizer
`__str__`	To string
`__hash__`	Get hash

Develop
Intelligence

Review

1. Name an advantage of using an object instead of a tuple or a dictionary
2. Explain the idea of 'dunder methods'
3. Define the Python dataclass construct