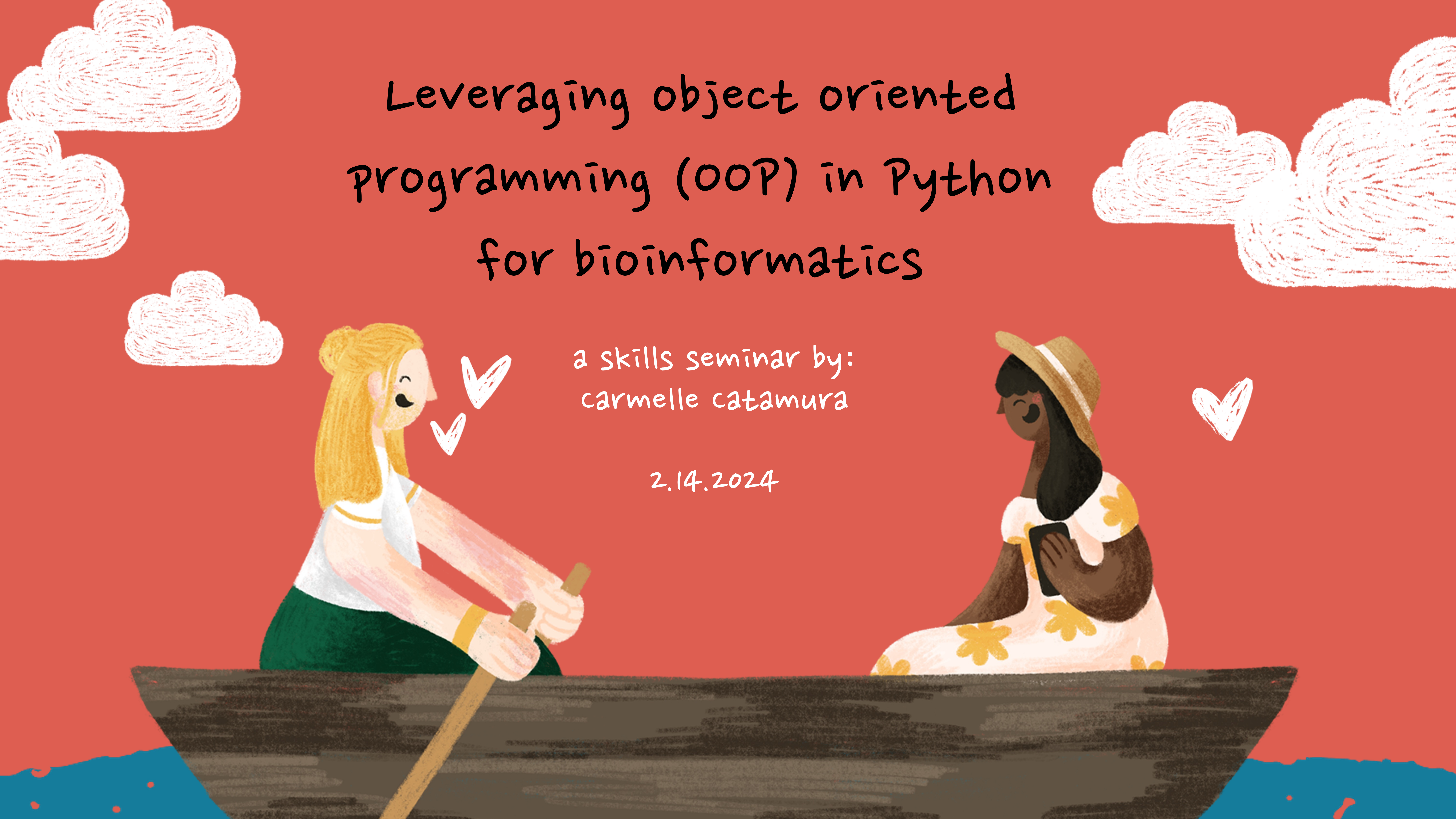
Leveraging object oriented programming (OOP) in Python for bioinformatics

a skills seminar by:
carmelle catamura

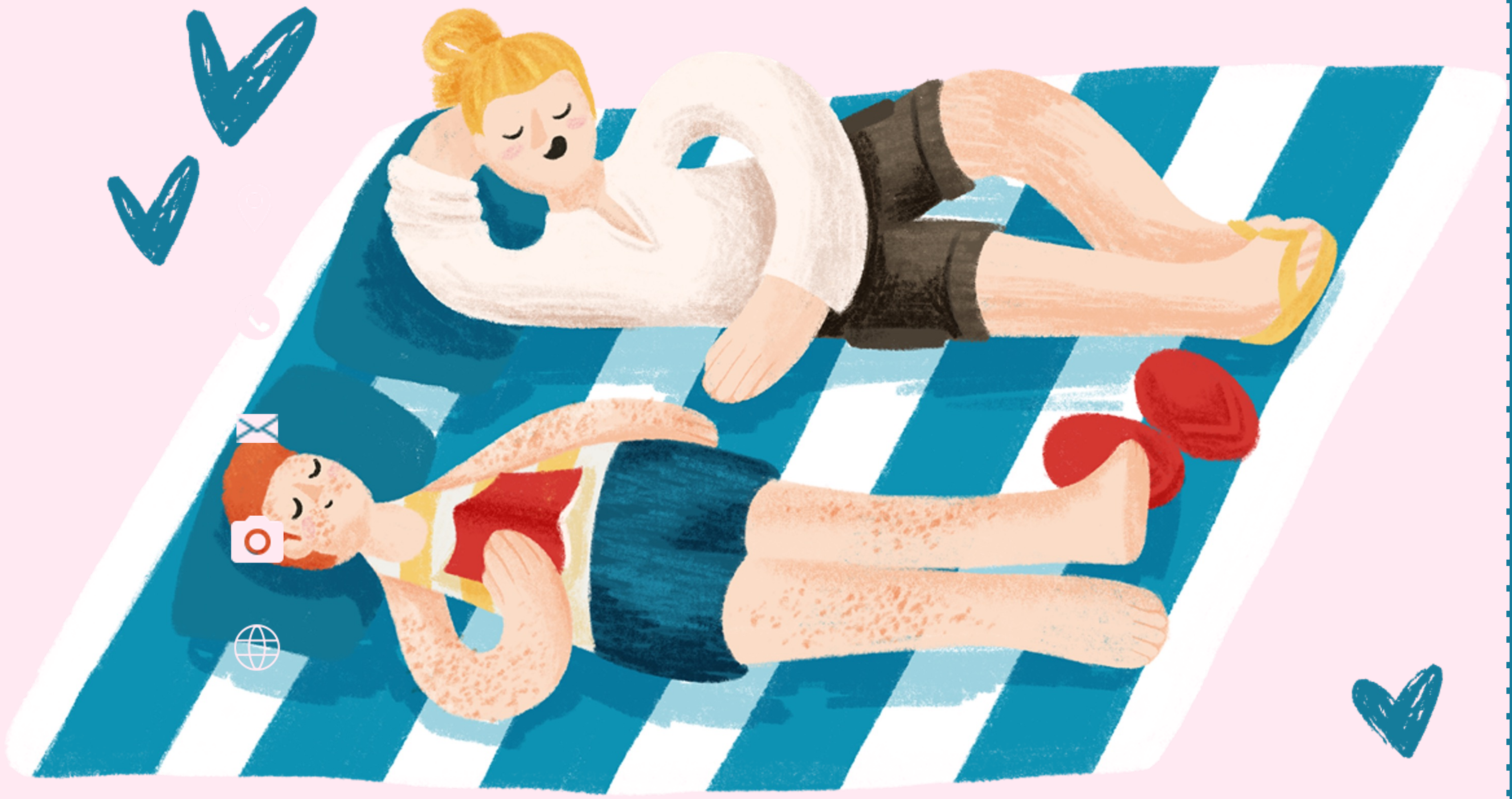
2.14.2024



What is OOP?

Object oriented programming (OOP) is a method of structuring your program so that data and functions that work together are organized into a single unit called an object.

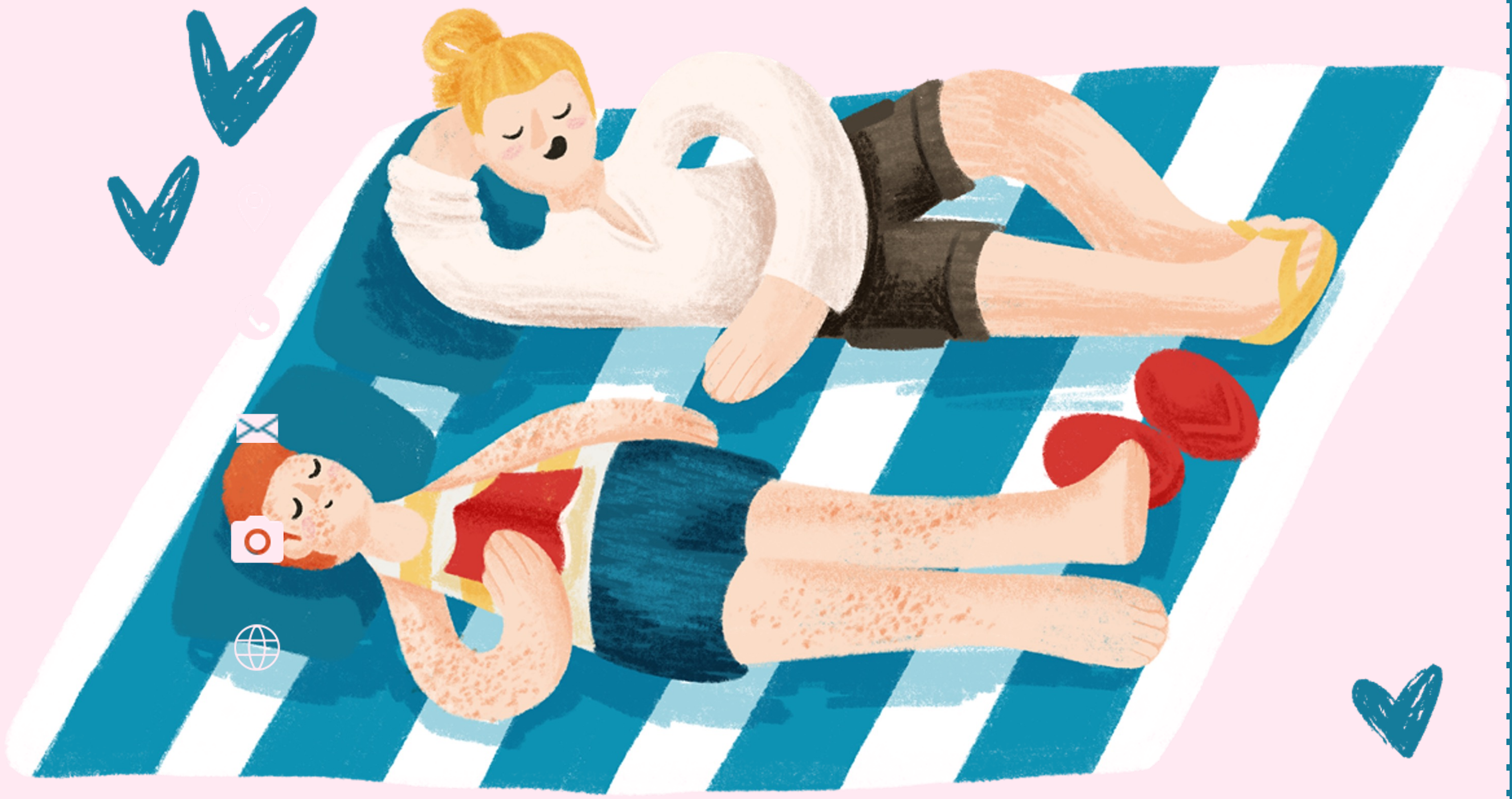
What are the benefits to OOP?



- better code organization
- easier code maintenance
- enforces modularity and reusability when writing code
- benefits of design - you can get really creative when solving problems

XOXO

What are the downsides to OOP?



- Sometimes, OOP is not the answer!
- "Simple is better than complex"
- "Practicality beats purity"

XOXO

what is a class?

a class is like a blueprint for creating a certain object.

what is a class?

a class is like a blueprint for creating a certain object.

what is an object?

an instance of a class

What is a class?

a class is like a blueprint for creating a certain object.

What is a method?

a method is a function belonging to the class

What is an object?

an instance of a class

What is a class?

a class is like a blueprint for creating a certain object.

What is an object?

an instance of a class

What is a method?

a method is a function belonging to the class

What is an attribute?

a data element or a property of an object

__init__()

every class needs this! initializes an object of a class; a constructor

self

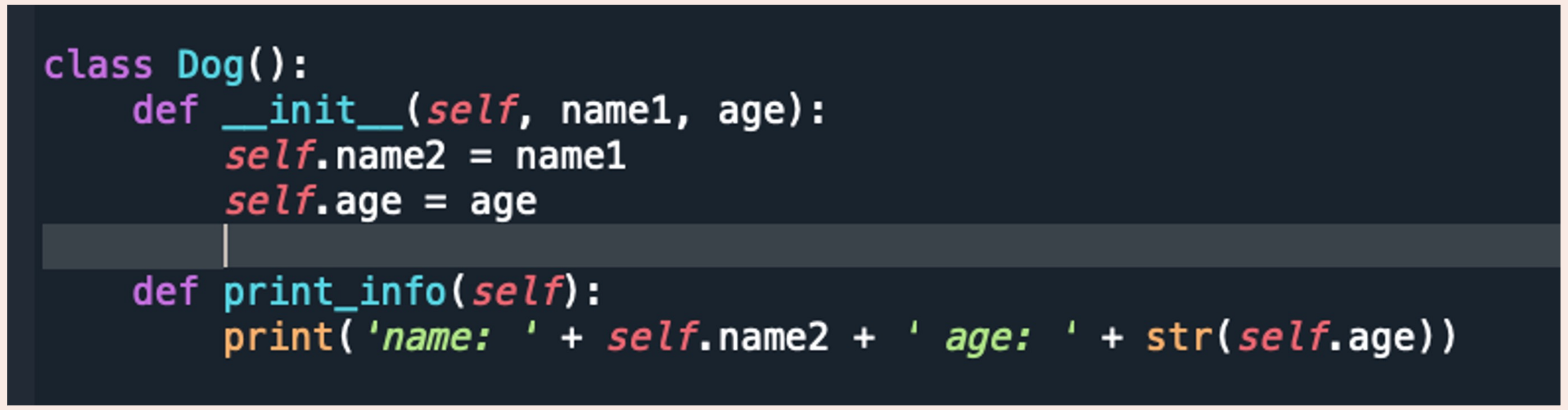
represents an instance of the class; the first parameter to all the methods in a class; by using self we can access attributes and methods belonging to an object.

xoxo

Let's try it!

Follow along using a text editor or a python IDE :)

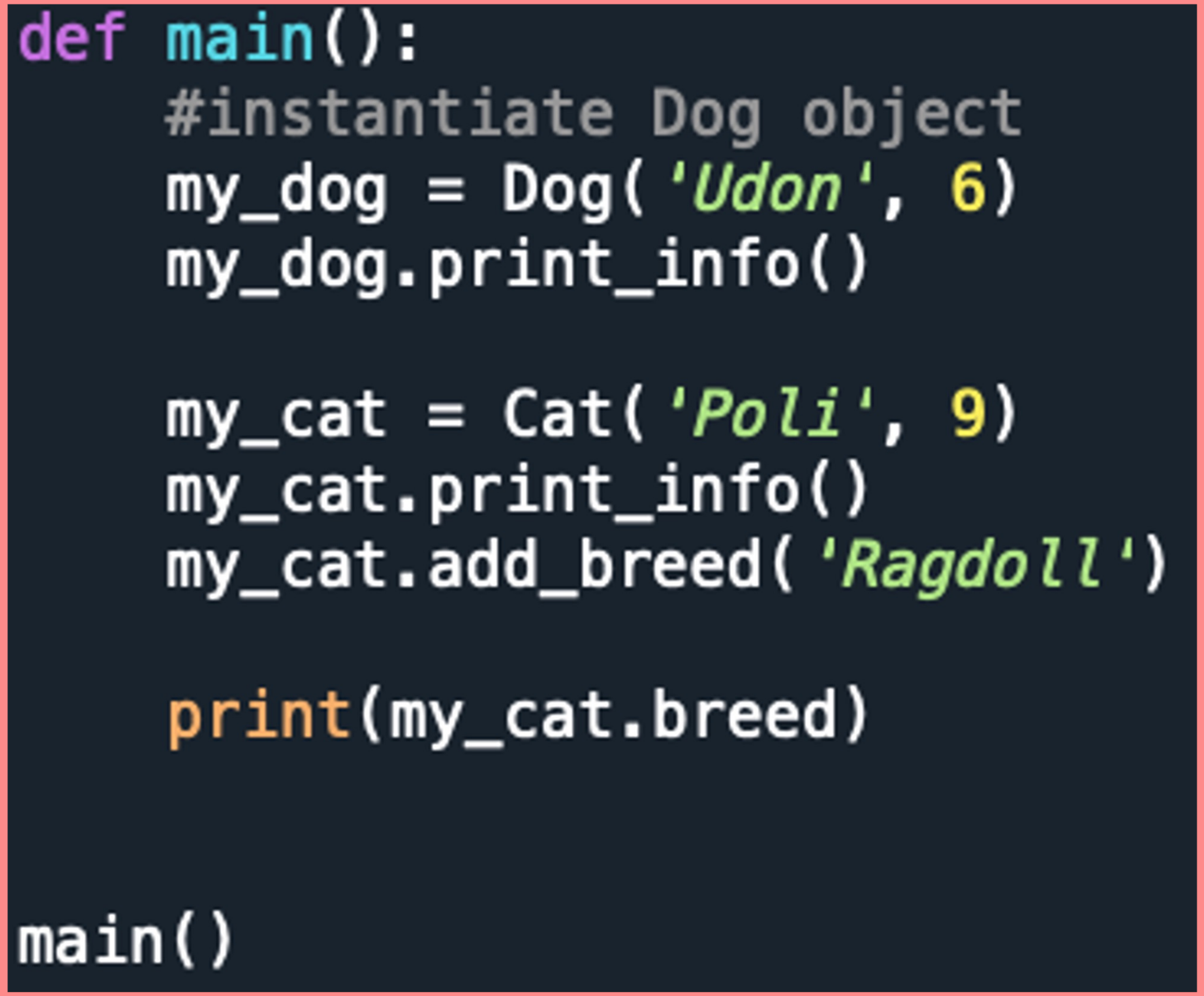
create a class called "Dog"



```
class Dog():
    def __init__(self, name1, age):
        self.name2 = name1
        self.age = age

    def print_info(self):
        print('name: ' + self.name2 + ' age: ' + str(self.age))
```

Accessing functions and properties of a class

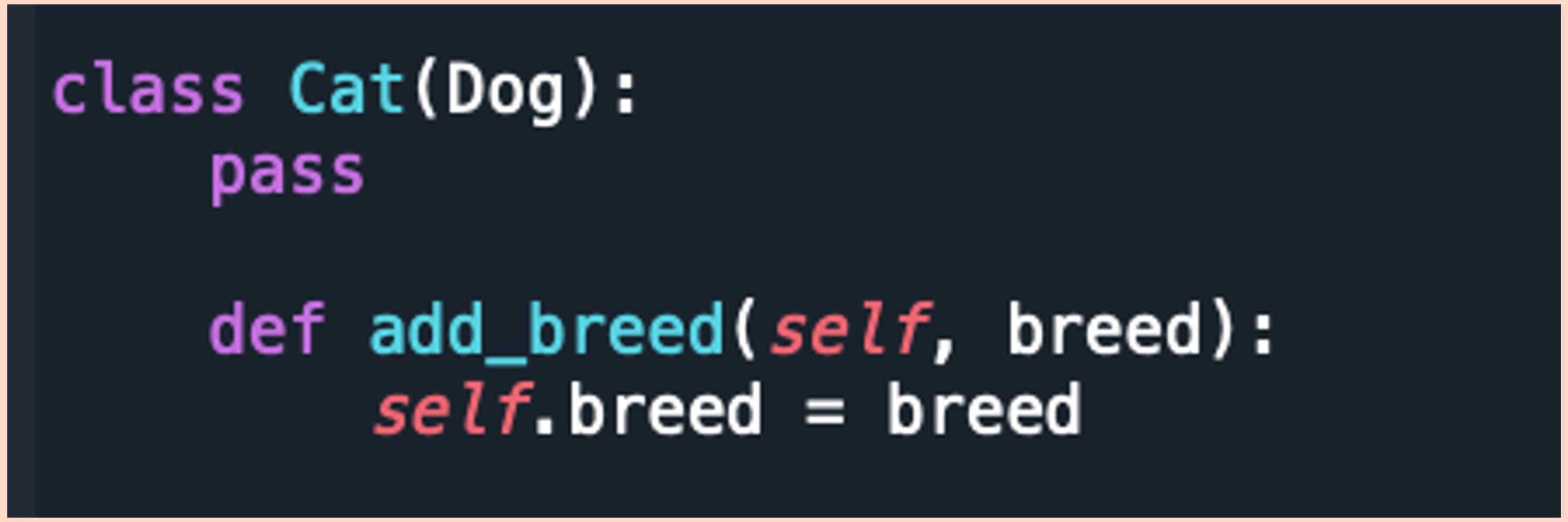
you can access properties and attributes of a class using . ('dot')



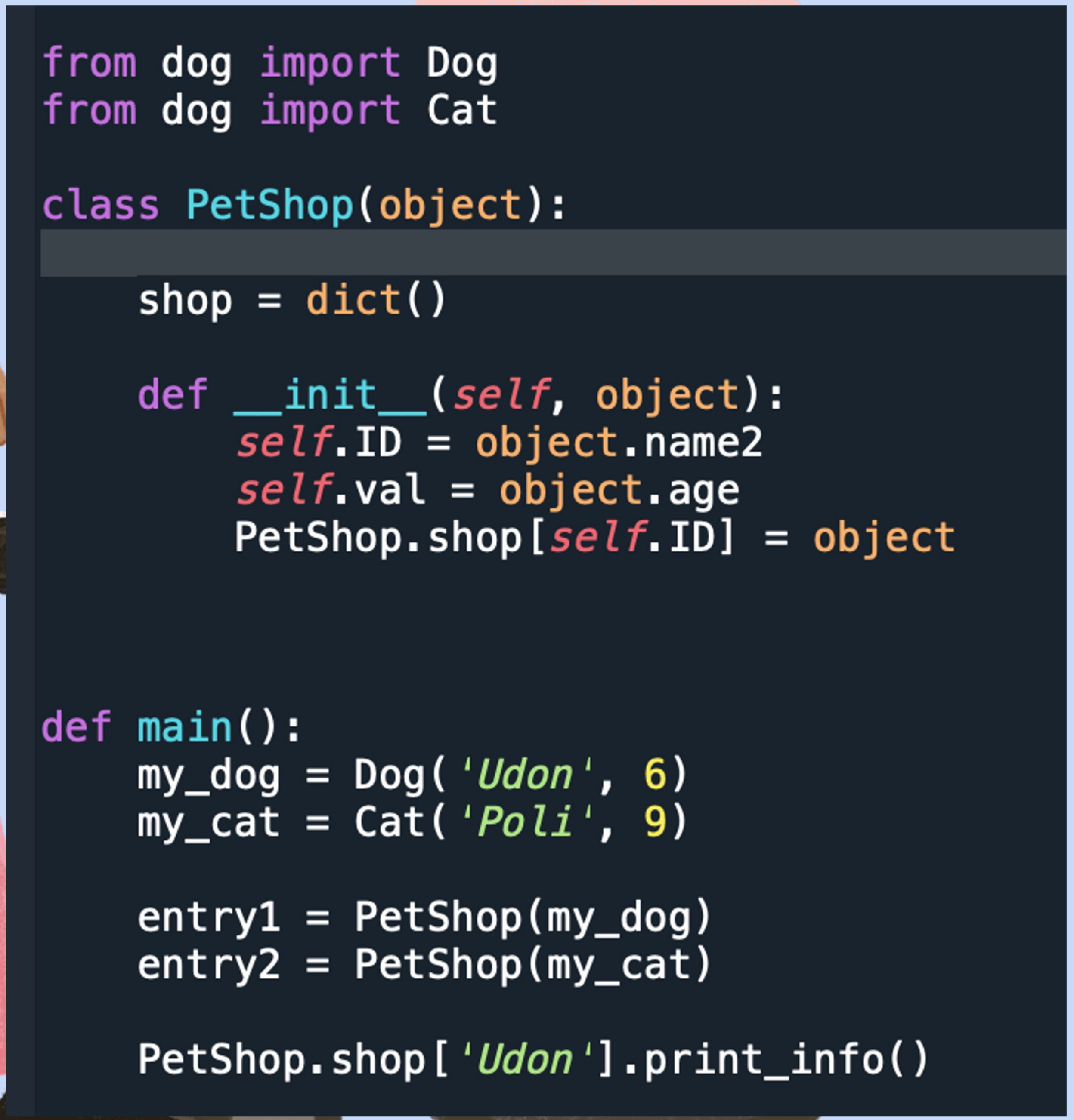
```
def main():
    #instantiate Dog object
    my_dog = Dog('Udon', 6)
    my_dog.print_info()

    my_cat = Cat('Poli', 9)
    my_cat.print_info()
    my_cat.add_breed('Ragdoll')

    print(my_cat.breed)


main()
```

class inheritance

a class can inherit all attributes and methods from another class



```
class Cat(Dog):
    pass

    def add_breed(self, breed):
        self.breed = breed
```

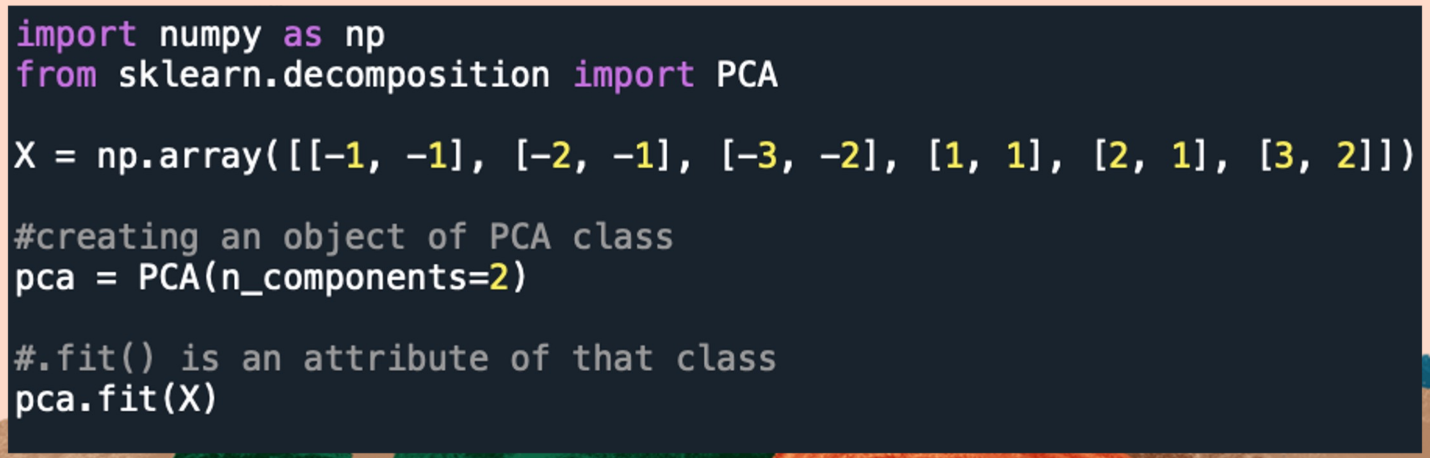


```python
from dog import Dog
from dog import Cat

class PetShop(object):

    shop = dict()

    def __init__(self, object):
        self.ID = object.name2
        self.val = object.age
        PetShop.shop[self.ID] = object



def main():
    my_dog = Dog('Udon', 6)
    my_cat = Cat('Poli', 9)

    entry1 = PetShop(my_dog)
    entry2 = PetShop(my_cat)

    PetShop.shop['Udon'].print_info()
```

You can easily import your class to another script!

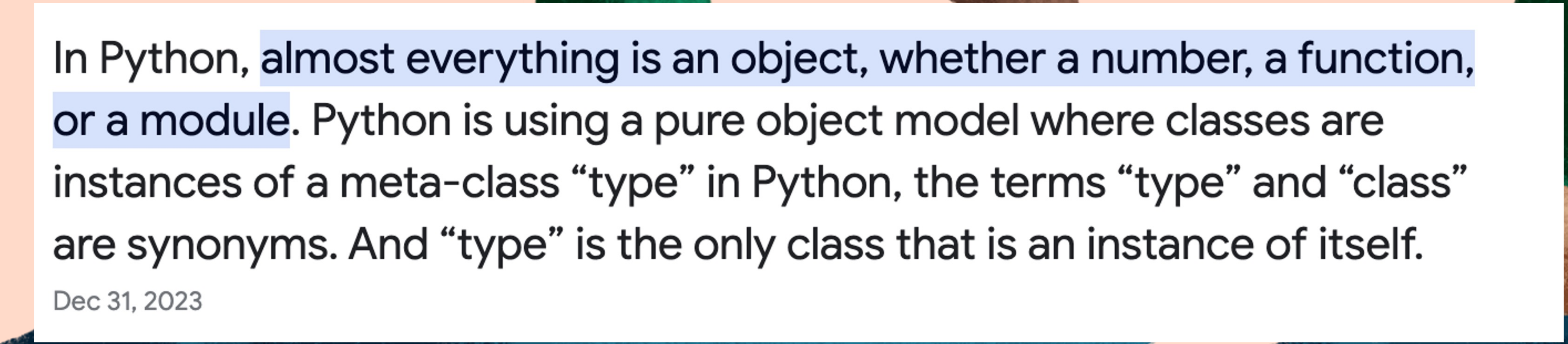
(What is Abstraction!?)

Everything in Python is an object! Lots of cool Python libraries are written OOP style



```
import numpy as np
from sklearn.decomposition import PCA

X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])

#creating an object of PCA class
pca = PCA(n_components=2)

#.fit() is an attribute of that class
pca.fit(X)
```

Everything in Python is an object! Lots of cool Python libraries are written OOP style



In Python, almost everything is an object, whether a number, a function, or a module. Python is using a pure object model where classes are instances of a meta-class “type” in Python, the terms “type” and “class” are synonyms. And “type” is the only class that is an instance of itself.

Dec 31, 2023

How I use OOP in my research IRL

Thank you!

if you would like a copy of the slides/scripts email me at carmelle@berkeley.edu