ISLab Python Course

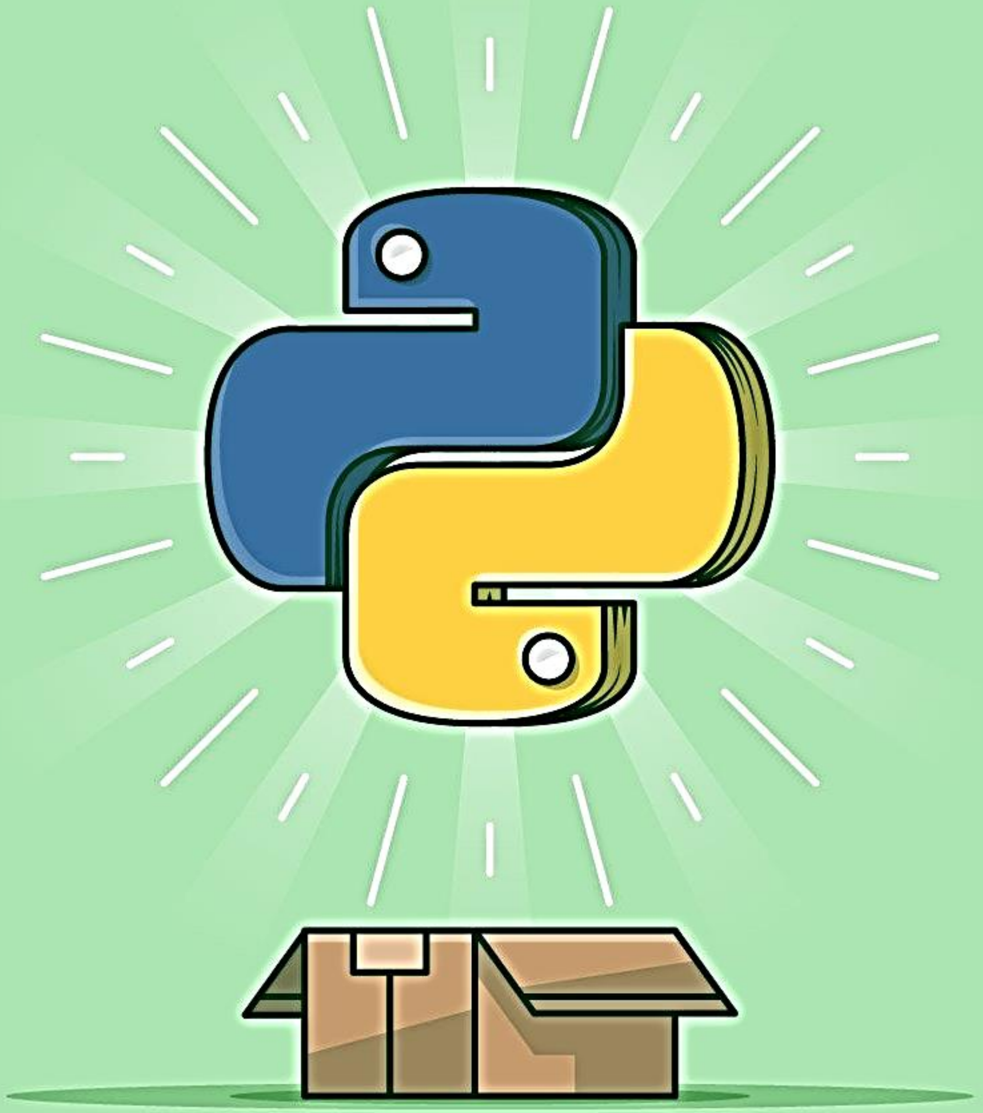
Session 8:
Object-Oriented Programming in Python

Presenters:

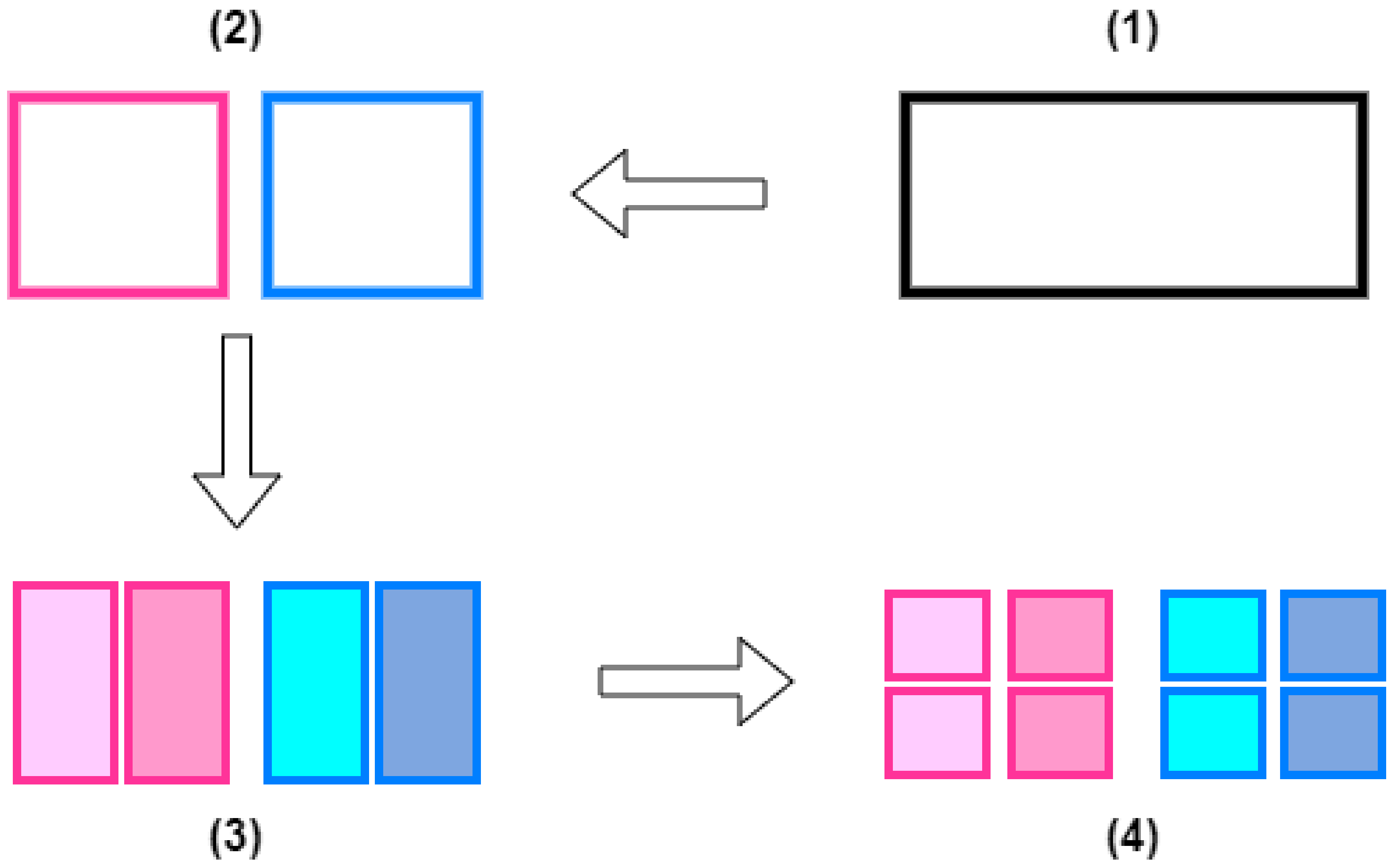
Shahrzad Shashaani   Hamed Homaei Rad   Saeed Samimi

Summer 2023

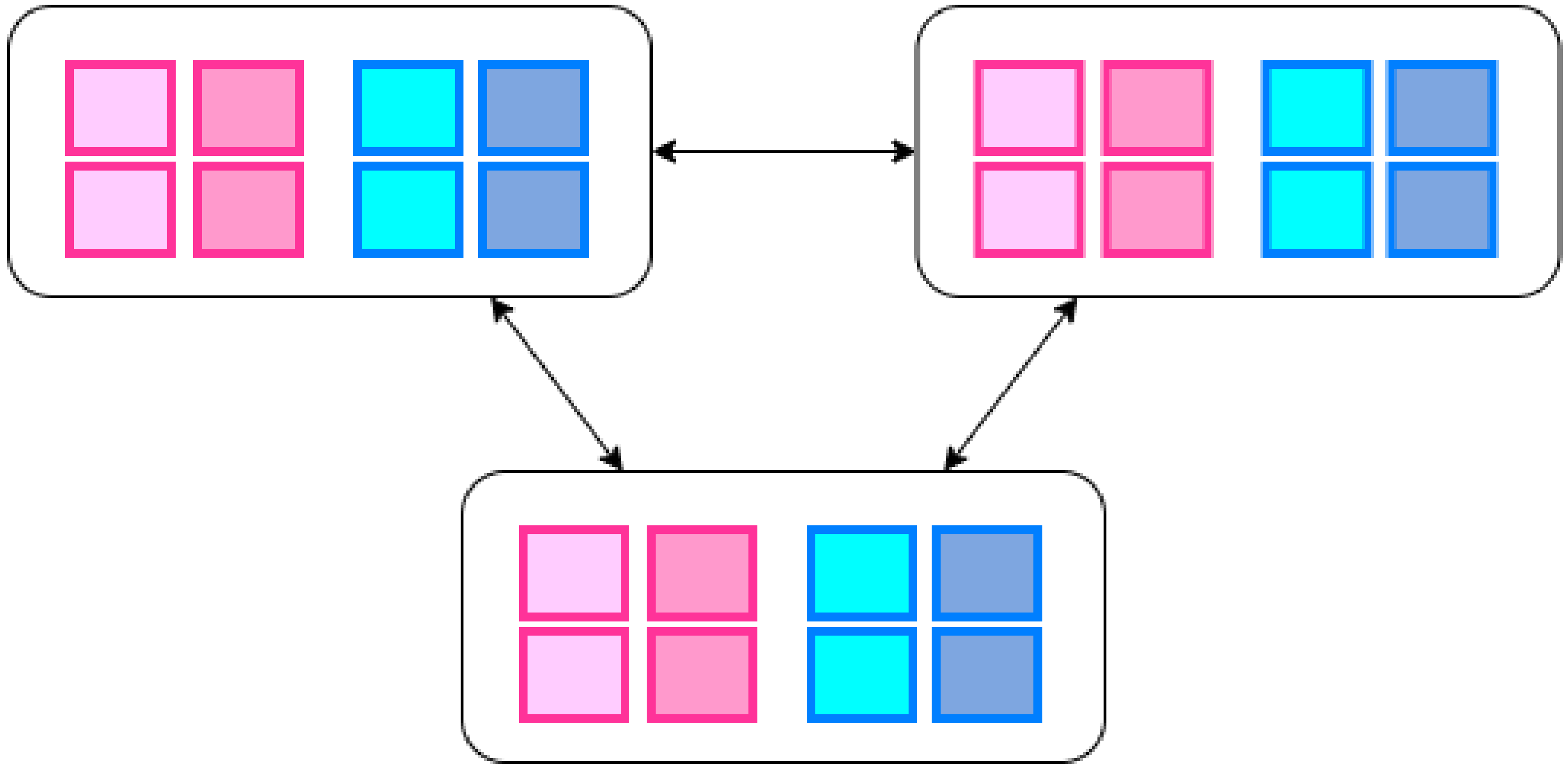
K.N.Toosi University of Technology



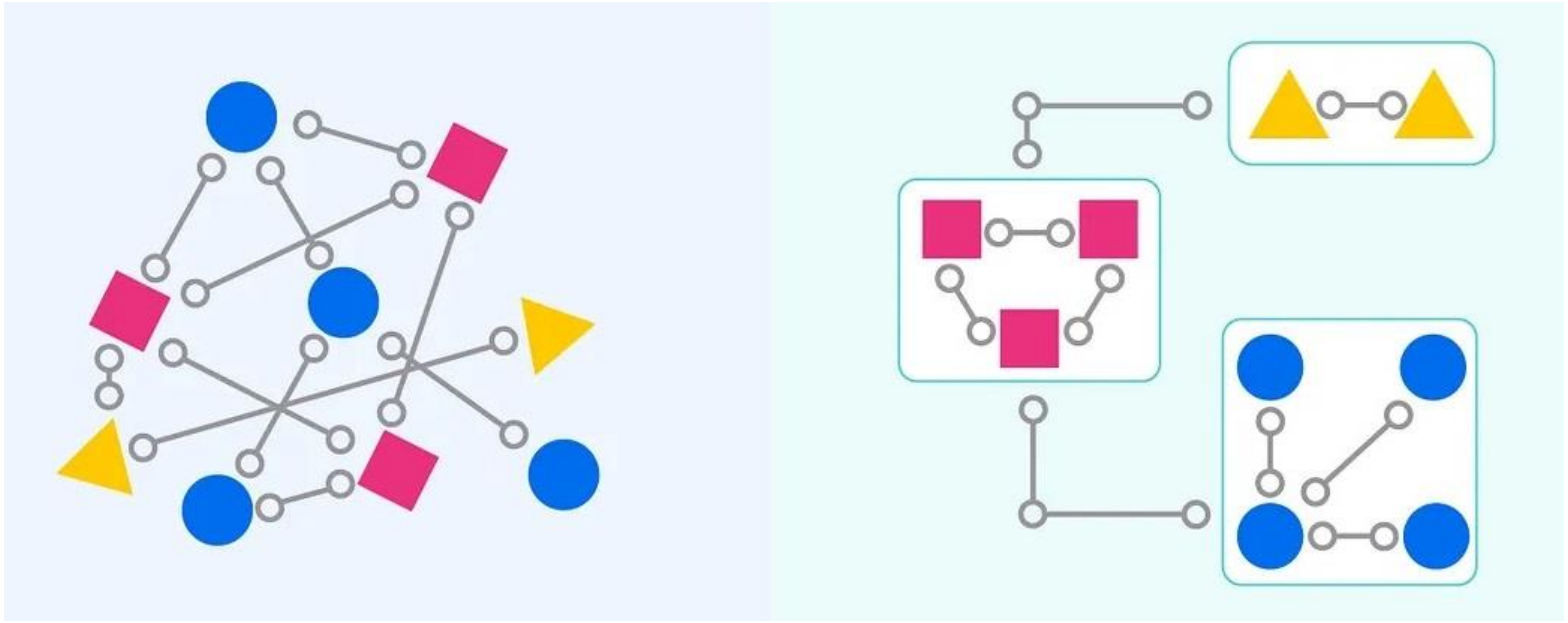
Breaking Down a Typical Program



Breaking Down a Typical Program



Separation of Concerns



https://medium.com/@onuryanar/separation-of-concerns-eca786e75f5a

Separation of Concerns

- **Modularity**
  - changes to one module should not affect other modules
  - => more robustness and flexibility
- **Reusability**
  - individual modules should address specific concerns
  - => we should be able to reuse them in different programs
- **Easier Debugging**
  - issues can be traced back to a specific module, rather than the entire program
  - => modules could go under extensive tests to ensure desired outcomes
- **Simpler Development Process**
  - Each developer focuses on a specific task and module development
  - => faster development process

Separation of Concerns

[Alexander Shvets] Dive Into Design Patterns (2019)

Separation of Concerns

Dictionary Array

```
new_club_member = {
    "name": "Mehrdad",
    "birth_date": "11/29/2001",
    "rating": 9.5,
    "memberships": ["gym", "pool", "aerobics"]
}
```

Function

```
def  some_fn(input1, input2):
    do_sth_with_input1
    do_sth_with_input2
    return sth
```

Separation of Concerns

Dictionary Array

```
new_club_member = {
    "name": "Mehrdad",
    "birth_date": "11/29/2001",
    "rating": 9.5,
    "memberships": ["gym", "pool", "aerobics"]
}
```

Function

```
def  some_fn(input1, input2):
    do_sth_with_input1
    do_sth_with_input2
    return sth
```

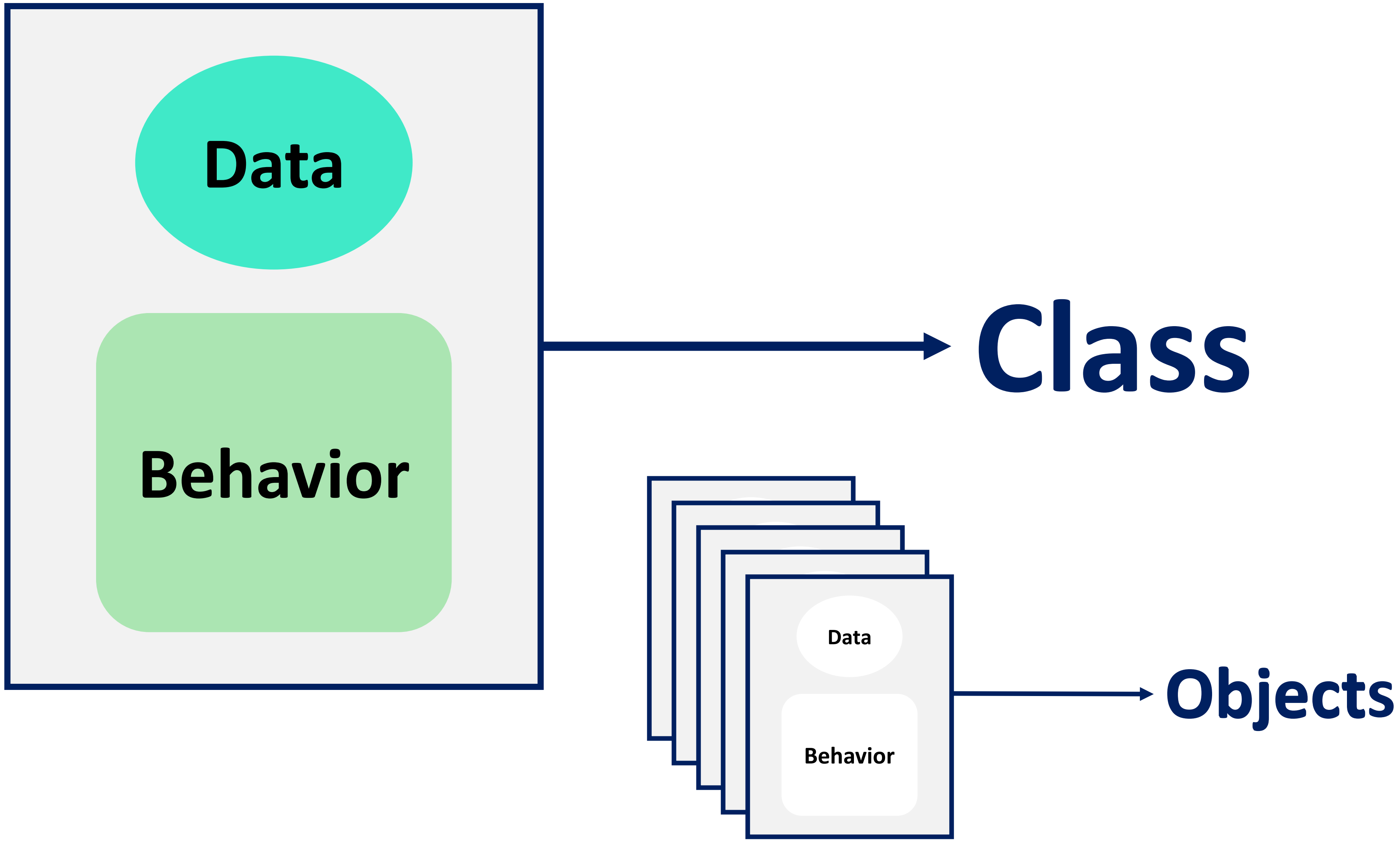
Data

(State)

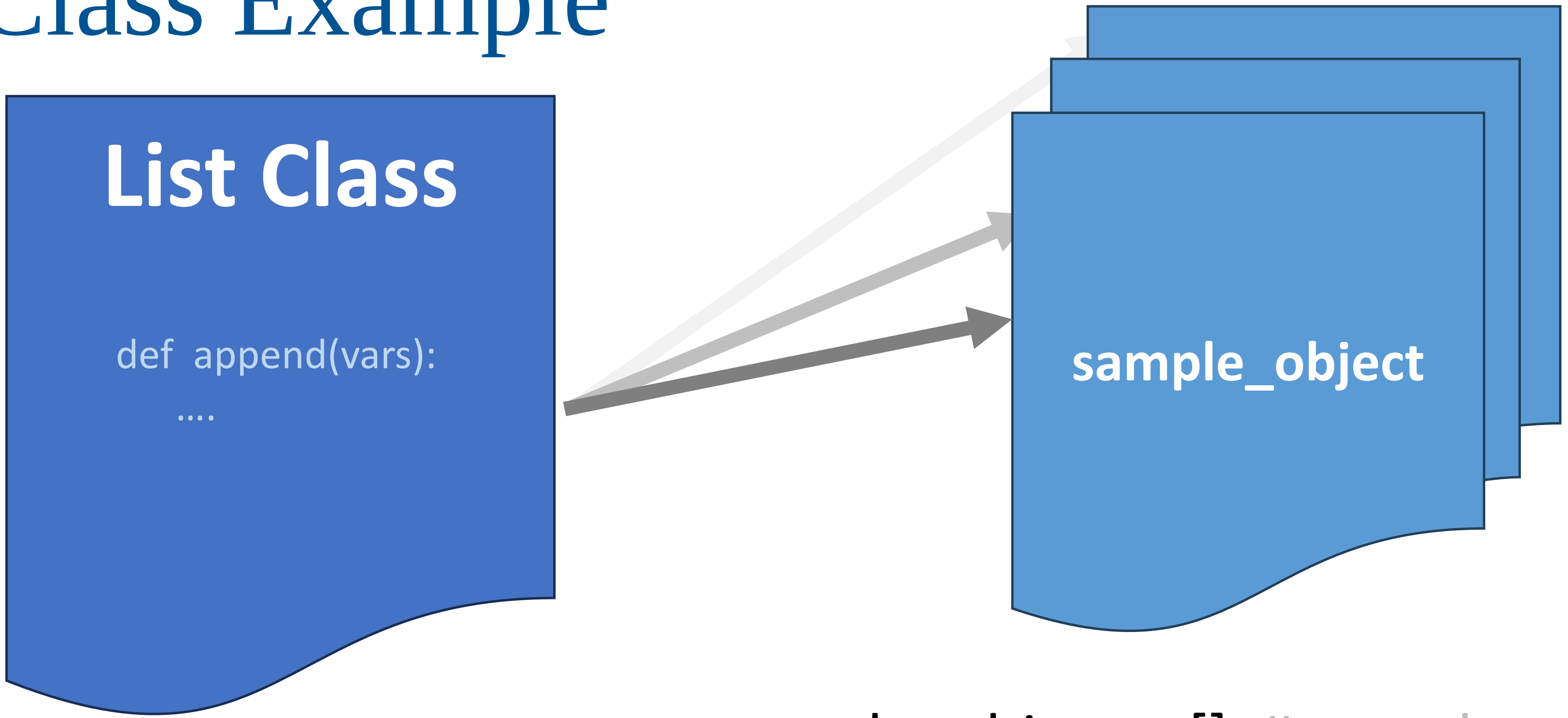
Actions

(Behavior)

What is OOP?



A Class Example



```
sample_object = []  # sample_object = list()
sample_object.append('python learners')
```

The Four Pillars of OOP

Encapsulation
- **Data Hiding**
- **Modularity**
- **Flexibility**

Abstraction
- Abstract Classes
- Interfaces

Inheritance
- **Parent Class:** Common Properties & Behaviors
- **Child Class:** Extending Properties & Behaviors

Polymorphism
- Method Overriding
- Method Overloading