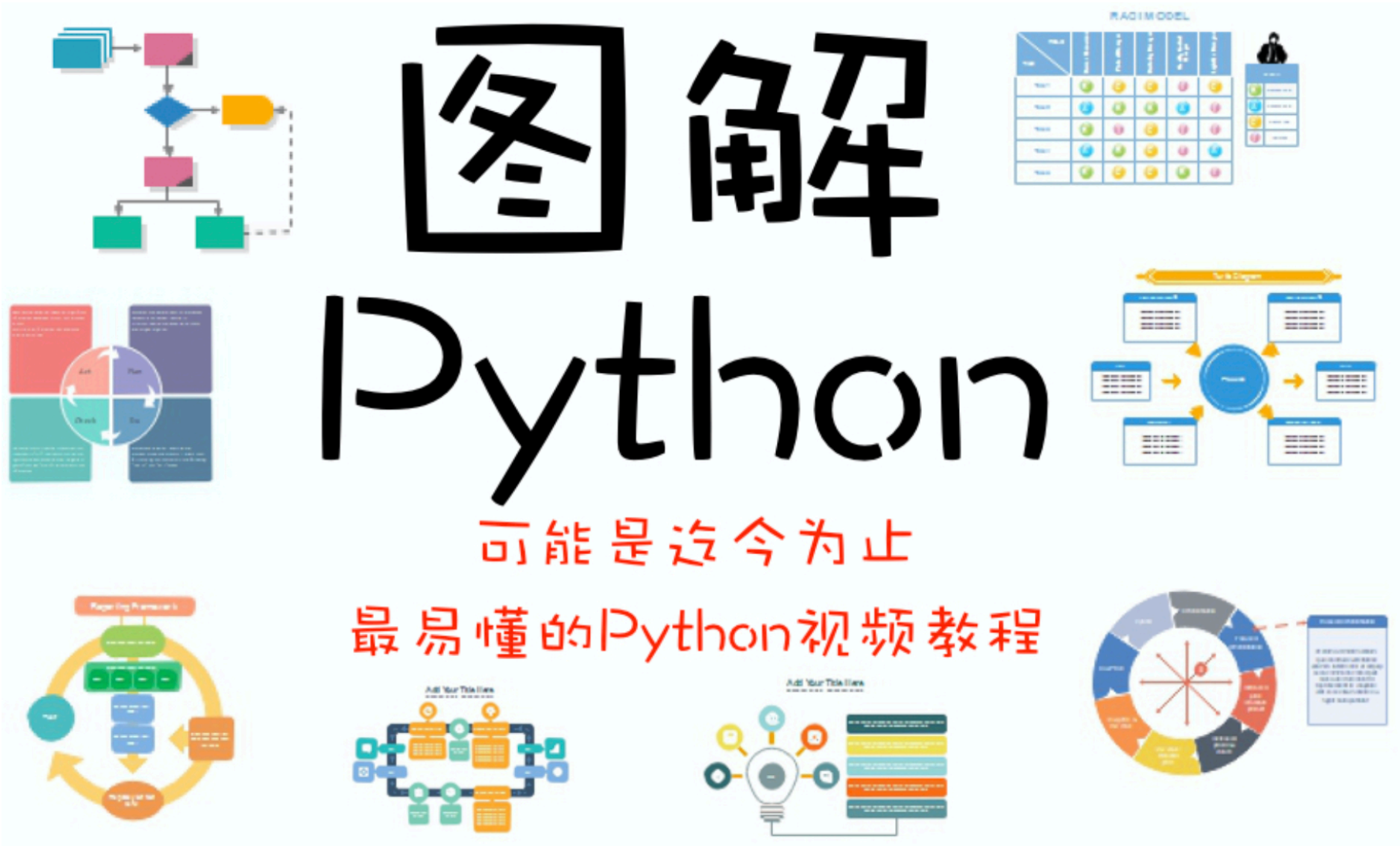
图解
Python
可能是迄今为止
最易懂的Python视频教程

```
"""类对象的特殊方法之__iter__()和__next__()"""

L = [1, 2, 3, 4, 5]

for item in L:
    print(item)

"""
    for-in语句在默认情况下不能用于自定义类对象的实例对象。
"""
# class MyClass(object):
    # pass

# for item in MyClass():  # TypeError: 'MyClass' object is not iterable
    # print(item)


"""
    如果想让for-in语句可以用于自定义类对象的实例对象，必须在自定义类对象中实现特殊方法
__iter__()和__next__()。for-in语句首先会调用特殊方法__iter__()返回一个可迭代对象，
然后不断调用可迭代对象的特殊方法__next__()返回下一次迭代的值，直到遇到StopIteration时退出循环。
    只实现了特殊方法__iter__()的类对象，被称为可迭代类对象；同时实现了特殊方法__iter__()和
__next__()的类对象，被称为迭代器类对象。
    之所以for-in语句可以用于某些内置类对象（例如：list、tuple、str等）的实例对象，
是因为这些内置类对象中都同时实现了特殊方法__iter__()和__next__()。
"""
```

```
class MyClass(object):
    def __init__(self):
        self.data = 0

    def __iter__(self):
        return self

    def __next__(self):
        if self.data > 5:
            raise StopIteration()
        else:
            self.data += 1
            return self.data

for item in MyClass():
    print(item)
```