# 面向对象作业题（04）

1. 类变量和示例变量的区别?
2. super的作用?
3. isinstance和type的区别并用代码举例说明?
4. 补全代码

```
def func(arg):
    """
    判断arg是否可以被调用，如果可以则执行并打印其返回值，否则直接打印结果
    :param arg: 传入的参数
    """
    pass
```

5. 补全代码

```
def func(*args):
    """
    计算args中函数、方法、Foo类对象的个数，并返回给调用者。
    :param args: 传入的参数
    """
    pass
```

6. 看代码写结果并画图表示对象和类的关系以及执行流程

```
class StarkConfig(object):
    list_display = []

    def get_list_display(self):
        self.list_display.insert(0,33)
        return self.list_display

class RoleConfig(StarkConfig):
    list_display = [11,22]

s1 = StarkConfig()
s2 = StarkConfig()

result1 = s1.get_list_display()
print(result1)

result2 = s2.get_list_display()
print(result2)
```

7. 看代码写结果并画图表示对象和类的关系以及执行流程

```
class StarkConfig(object):
    list_display = []
```

```
    def get_list_display(self):
        self.list_display.insert(0,33)
        return self.list_display

class RoleConfig(StarkConfig):
    list_display = [11,22]

s1 = StarkConfig()
s2 = RoleConfig()

result1 = s1.get_list_display()
print(result1)

result2 = s2.get_list_display()
print(result2)
```

8. 看代码写结果并画图表示对象和类的关系以及执行流程

```
class StarkConfig(object):
    list_display = []

    def get_list_display(self):
        self.list_display.insert(0,33)
        return self.list_display

class RoleConfig(StarkConfig):
    list_display = [11,22]

s1 = RoleConfig()
s2 = RoleConfig()

result1 = s1.get_list_display()
print(result1)

result2 = s2.get_list_display()
print(result2)
```

9. 看代码写结果

```
class Base(object):
    pass

class Foo(Base):
    pass

print(issubclass(Base, Foo))
```

10. 背写你了解的所有特殊方法并附示例

11. 如果有以下handler.py文件，请在run.py中补充代码实现：

- 获取handler中所有成员名称：dir(handler)
- 获取handler中名字叫Base的成员
- 检查其他成员是否是Base类的子类(不包含Base)，如果是则创建对象并添加到objs列表中。

```
# handler.py
class Base(object):
    pass

class F1(Base):
    pass

class F2(Base):
    pass

class F3(F2):
    pass

class F4(Base):
    pass

class F5(object):
    pass

class F6(F5):
    pass

# run.py
import handelr
def func():
    objs = []
    name_list = dir(handelr)
    print(name_list)

if __name__ == '__main__':
    func()
```

## 12. 补充代码

```
class Foo(object):
    def __init__(self):
        self.name = '武沛齐'
        self.age = 100
obj = Foo()
setattr(obj, 'email', 'wupeiqi@xx.com')
```

请实现：一行代码实现获取obj中所有成员（字典类型）

## 13. 看代码写结果

```
class Foo(object):
```

```
    def __init__(self):
        self.name = '武沛齐'
        self.age = 100

obj = Foo()
setattr(Foo, 'email', 'wupeiqi@xx.com')

v1 = getattr(obj, 'email')
v2 = getattr(Foo, 'email')

print(v1,v2)
```

14. 什么是可迭代对象？如何将一个对象编程可迭代对象？
15. 如何让一个对象可以被执行？即：后面加()

16. 补充选课系统（必须用反射）

```
class Course:
    def __init__(self,name,price,period):
        self.name = name
        self.price = price
        self.period = period

class Student:
    def __init__(self,name):
        self.name = name
        self.courses = []

    def select_course(self):
        """选择课程，已选则不能再选"""
        pass

    def show_selected_course(self):
        """查看自己所选课程"""
        pass
    def show_selected_course(self):
        """删除已选课程"""
        pass
def run():
    """
    主程序
        1. 根据Course类创建10个课程
        1. 用户输入学生姓名，动态创建学生对象。
        2. 查看所有课程
        3. 为学生选课
        4. 查看学生已课程
        5. 删除已选课程
    """
    pass
```