问答题

1） 从静态特征和动态特征两个方面对'病毒'对象进行描述

2） 列举你了解到的 10 个类的专有方法，具体描述其功能，返回值，参数，并举例说明。

解答：

第一题：

- 静态特征：结构简单，只包含DNA或者RNA，拥有蛋白质外壳
- 动态特征：没有代谢机构，无法独立自我繁殖，需寄生在其他细胞上

第二题：

1. ```
   __init__(self):
   ```

   - 功能：初始构造方法，初始化类内变量的值
   - 返回值：无
   - 参数：至少有一个self变量。
   - 例子：

   ```
   class Car:
       __wheel = 4
       __color = 'Pearl White'
       def __init__(self, wh_num, co):
           __wheel = wh_num
           __color = co
           print('----调用了构造方法----')

   # 上面代码输出结果为：-----调用了构造方法-----
   ```

2. ```
   __del__(self):
   ```

   - 功能：析构方法，删除类时会进行的操作
   - 返回值：无
   - 参数：一个self变量
   - 例子：

   ```
   class Car:
       def __del__(self):
           print('----调用了析构方法----')

   # 上面代码输出结果为：-----调用了析构方法-----
   ```

3. ```
   __str__(self, args):
   ```

   - 功能：把一个类的实例变成str
   - 返回值：字符串
   - 参数： 一个self变量
   ```

- 例子：

```
class Student:
    def __init__(self,name):
        self.name = name

    def __str__(self):
        return "学生姓名：%s" % self.name

s = Student("xiaoh")
print(s)

# 上面代码的输出结果为：学生姓名：xiaoh
```

4.
```
__call__(self, args):
```

- 功能：直接对实例进行调用
- 返回值：类中实参
- 参数：至少有一个self变量
- 例子：

```
class Student:
    def __init__(self):
        self.name = "xiaohong"

    def __call__(self,score):
        self.score = score
        print("姓名：%s,分数：%s" % (self.name,self.score))


student = Student()
student(66)

"""
上面代码的输出结果为：
姓名：xiaohong,分数：66
"""
```

5.
```
__iter__(self):
```

- 功能：在类中生成一个迭代器
- 返回值：无
- 参数：一个self变量
- 例子：(与6放在一起使用)

6.
```
__next__(self):
```

- 功能：决定迭代的规则
- 返回值：类中迭代器中的变量的值
- 参数：一个self变量
- 例子： (与5一起使用)

```
class counter:
    def __init__(self):
```

```
        self.a = 0
        self.b = 1


    def __iter__(self):
        return self

    def __next__(self):
        self.a, self.b = self.b, self.a + self.b
        if self.a >10:
            raise StopIteration
        return self.a
for number in Fib():
    print(number)

# 上面代码的输出结果为：1、1、2、3、5、8
-----------------
原文链接：https://blog.csdn.net/qq_39314932/article/details/81088784
```

7.
```
__getitem__(self, item):
```

- 功能：像list一样按照索引来获取元素
- 返回值：迭代器中的值
- 参数：2个，一个self参数和一个下标数
- 例子：

```
class Fib:
    def __init__(self):
        self.a = 0
        self.b = 1

    def __iter__(self):
        return self

    def __next__(self):
        self.a ,self.b = self.b,self.a + self.b
        return self.a

    def __getitem__(self, item):
        a = 1
        b = 1
        for num in range(item):
            a, b = b, a + b
        return a

if __name__ == "__main__":
    fib = Fib()
    for number in fib:
        if number > 10:
            break
        print(number)
    print(fib[5])

# 上面代码的输出结果为：1、1、2、3、5、8、8
```

8.
```
__getattr(self, args):
```

- 功能：当python类调用不存在的属性时，python就会调用`__getattr__( )`尝试获得属性，这样就会返回对应的属性
- 返回值：实参
- 参数，至少2个，一个self参数和一个变量
- 例子：

```
class Student:
    def __init__(self):
        self.name = "xiaohong"

    def __getattr__(self,attr):
        if attr == "score":
            return 95

student = Student()
print(student.name)
print(student.score)

# 上面代码的输出结果为：
# xiaohong
# 95
```

9.
```
delattr(object, name):
```

- 功能：删除类中的属性
- 返回值：无
- 参数：类名和对象属性名
- 例子：

```
class A(object):
    bar = 1
    bar_1 = "2"

a = A()
delattr(A,"bar")  #为类名
delattr(a,"b")

print(a.bar)    #属性已被删除时的输出
print(a.bar_1)  #无对应属性时的输出

"""
上面代码的输出结果为：
#AttributeError: b
#'A' object has no attribute 'bar'
"""
```

10.
```
getattr(object, name[, default]):
```

- 功能：返回一个对象指定的属性值。如果指定的属性不存在，则返回default的值，若没有设置default参数，则抛出AttributeError异常。
- 返回值：对象属性值

- 参数：3个，对象，对象属性（字符串类型），默认返回值
- 例子：

```
class A(object):
    bar = 1

a = A()
if getattr(a,"bar","我是默认的属性") == 1:
    print("bar的属性为%s" % 1)

attr_no = getattr(a,"b","我是默认的属性")
print("no的属性为%s" %(attr_no))

"""
上面代码的输出结果为：
bar的属性为1
no的属性为我是默认的属性
"""
```

上机题

请用 password 类来实现 PPT 中测试密码强度的程序。

源代码：

```
class Password:
    pw = 'None'  # 密码
    intensity = 0  # 密码强度等级
    digit = False
    alphas = False

    def __init__(self, p):
        self.pw = p

    def judge(self):
        if len(self.pw) >= 8:
            self.intensity = 1  # 符合最低要求的密码强度为1
        else:
            print('密码长度小于8个字符！')
            while len(self.pw) < 8:
                self.pw = input('请设置密码（至少8个字符）：')

        pw_list = list(self.pw)
        for i in range(len(pw_list)):
            pw_list[i] = ord(pw_list[i])

        for i in range(len(pw_list)):
            if pw_list[i] in range(48, 58):
                self.digit = True  # 字符串中含有数字
                print('密码中含有数字')
                break

        for i in range(len(pw_list)):
```

```
            if pw_list[i] in range(65, 91) or pw_list[i] in range(97, 123):
                self.alphas = True  # 字符串中含有字母
                print('密码中含有字母')
                break


p_in = input('请设置密码（至少8位，仅包含数字和字母）：')
p_w = Password(p_in)
p_w.judge()
```

运行结果：