Python与深度学习基础

张越一　　zhyuey@ustc.edu.cn

关于本课

- 助教与QQ群
- 上次PPT的勘误
- 关于Anaconda
- 关于Python IDE (Pycharm)
- 关于Jupyter
  - Kaggle
    - Kaggle.com
  - Google Research Colab
    - https://colab.research.google.com

函数

- 函数是组织好的，可重复使用的，用来实现单一，或相关联功能的代码段。
- 函数能提高应用的模块性，和代码的重复利用率。
- 内建函数

		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

函数

- 内建函数，可以直接使用
- 举例：

```
>>> abs(-3.14)
3.14
>>> int(5.9)
5
```

```
>>> hex(25)
'0x19'
>>> bin(25)
'0b11001'
>>> oct(25)
'0o31'
>>>
```

括号一定要是半角的！！！

函数

- 内建函数
- print 打印信息
- help 获得帮助信息
- isinstance 判断是否为某种类型
- type 显示变量类型
- len 获取长度
- input 获取一个输入

函数

```
>>> namelist = ["Alpha", "Bravo", "Camp"]
>>> agelist = [35, 46, 78]
>>> heightlist = [1.78, 1.67, 1.89]
>>> for (name, age, height) in zip(namelist, agelist, heightlist):
...     print(name, age, height)
...
Alpha 35 1.78
Bravo 46 1.67
Camp 78 1.89
```

语法糖 in：

print(35 in [35, 67])

函数

- 自建函数
- 用def 起头，后接函数名，后接参数列表，后接冒号
- 函数体要缩进，函数的返回值用return 返回
- 位置参数与关键字参数　关键字参数要是位置参数的右边
- 举例:
- def my_mul(x, y):
  - return (x + 2) * y
- print(my_mul(3, 4))
- print(my_mul(x = 3, y = 4))
- print(my_mul(y = 4, x = 1))

```
>>> def my_mul(x, y):
...     return (x + 2) * y
...
>>> my_mul(3, 4)
20
>>> my_mul(x = 3, y = 4)
20
>>> my_mul(y = 4, x = 1)
12
>>>
```

函数

- 默认参数 （位置参数在前，默认参数在后）
- def my_mul(x, y = 2):
  - return (x + 2) * y
- print(my_mul(3))

- 可变参数 args, kargs
- 传入的参数数量，类型可变
- 参见 https://www.cnblogs.com/bingabcd/p/6671368.html

函数

- 多个返回值
- def mul_ret(x, y):
  - return 3*x, 4*y
- print(mul_ret(3, 4))

- 返回一个tuple
- 返回值不可更改

```
>>> rret = mul_ret(3, 4)
>>> print(rret)
(9, 16)
>>> type(rret)
<class 'tuple'>
>>> rret[0] = 1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>>
```

递归函数

- F(1)=1，F(2)=1, F(n)=F(n-1)+F(n-2)（n>=3，n∈N*）
- def fi(x):
  - if x in [1, 2]:
    - return 1
  - else:
    - return fi(x-1) + fi(x-2)
- print(fi(10))

```
>>> def fi(x):
...     if x in [1, 2]:
...         return 1
...     else:
...         return fi(x - 1) + fi(x-2)
...
>>> fi(10)
55
>>> fi(20)
6765
>>>
```

这不是一个好的写法，
笔面试时千万不要用。

重要事情说三遍

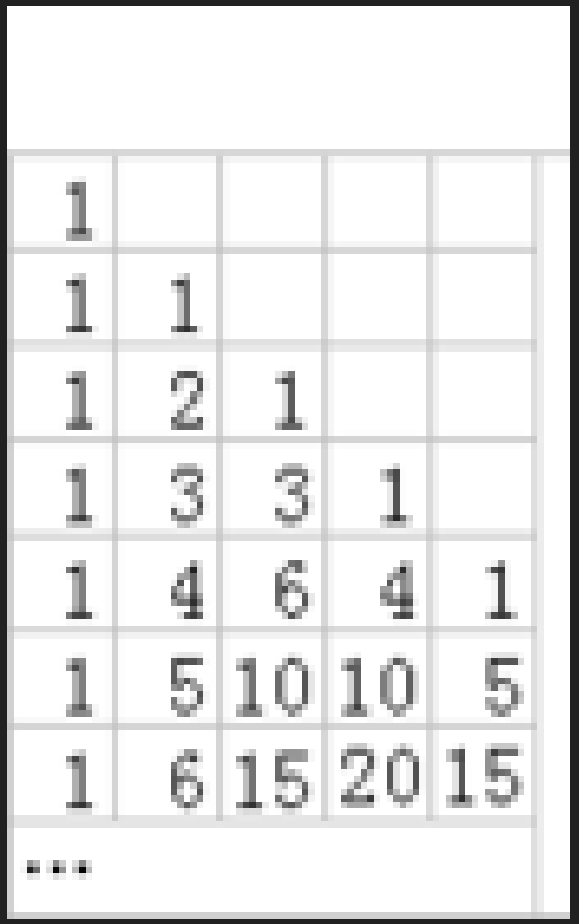
- 冒号
- 缩进
- 半角字符

- 冒号
- 缩进
- 半角字符

- 冒号
- 缩进
- 半角字符

课后小练习

- 杨辉三角 贾宪三角
- 写一个函数，求第i行，第j列的值
- 举例：
- fun(0, 0) = 1
- fun(5, 2) = 10



Leetcode

- LeetCode简介
- 可以使用LeetCode练习Python

面向对象

- Python是面向对象的编程语言
- 何为面向对象？
- 对象：进行研究的任何事物，也包括抽象的规则、计划与事件
- 对象的状态：数据
- 对象的行为：操作
- 对象是状态与行为的结合，是数据与操作的封装
- 类：具有相同特性和行为的对象的抽象

类

- class ClassName:
  - ‘class help information’ # 类文档字符串
  - class_suite　#类的主体
- 类里面加属性和函数
  - __init__ 类比于构造函数
  - __del__ 类比于析构函数 （不是本课重点）
- 高级话题：
- 有兴趣的同学，可以去学习Python的垃圾回收机制

类

- 举例：
- class Fruit:
- 　　def __init__(self, name, price):
- 　　　　self.name = name　　#也可以写在__init__函数外面
- 　　　　self.price = price
-
- 　　def eatIt(self):
- 　　　　print("Give me " + str(self.price) + " dollars!")
- 　　　　print("Let's enjoy " + self.name)

- apple = Fruit('apple', 3.5)
- apple.eatIt()

```
In [10]: runfile('C:/Users/zhyue/.spyder-py3/temp.py',
wdir='C:/Users/zhyue/.spyder-py3')
Give me 3.5dollars
Let's enjoy apple
```

类的继承

- 好处：代码重用
- 通过继承创建的新类称为**子类**或**派生类**，被继承的类称为**基类、父类**或**超类**。

```
class people:
    #定义基本属性
    name = ''
    age = 0
    #定义私有属性,私有属性在类外部无法直接进行访问
    __weight = 0
    #定义构造方法
    def __init__(self,n,a,w):
        self.name = n
        self.age = a
        self.__weight = w
    def speak(self):
        print("%s 说: 我 %d 岁。"
              %(self.name,self.age))
```

```
class student(people):
    grade = ''
    def __init__(self,n,a,w,g):
        #调用父类的构函
        people.__init__(self,n,a,w)
        self.grade = g
    #覆写父类的方法
    def speak(self):
        print("%s 说: 我 %d 岁了，我在读 %d 年级"
              %(self.name,self.age,self.grade))
```

万类之始

- object类
- Python中所有类的基类
- 如果没特意标明父类，默认继承object类

- dir(object)
- issubclass(float, object)

类的小练习

- 写一个分数类，支持自动约分
- class Frac

- a = Frac(2, 6)
- print(a)
- output: “Frac <1, 3>”

- Hint: __str__ 方法 __repr__ 方法

异常处理

- a = 10
- b = 0
- c = a / b
- print(“done”)

```
>>> a = 10
>>> b = 0
>>> a / b
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>>
```

异常处理

- a = 10
- b = 0
- try:
  - c = a/b
  - print(c)
- except ZeroDivisionError as e:
  - print(“Error： “, e)
- print(“done”)

自行学习

raise, finally

文件读写

- 读文件
- f = open(“text.txt”, “r”)
- print(f.readlines())
- f.close()

- 上面的代码可能有问题
- 最好加异常处理

- 推荐方法，用with
- with open(‘text.txt’, ‘r’) as f:
  - print(f.read())

- 相关函数
- read()
- readline()
- readlines()

文件读写

- 写文件
- with open('text.txt', 'w') as f:
  - f.write('Hello world')

- write()
- writelines()

- mode:
- r
- w
- a
- r+
- w+
- a+
- 二进制 b

文件读写

- csv 文件　python原生支持

```
LOW,AGE,LWT,RACE,SMOKE,PTL,HT,UI,BWT
1.0,28.0,113.0,1.0,1.0,1.0,0.0,1.0,709.0
1.0,29.0,130.0,0.0,0.0,0.0,0.0,1.0,1021.0
1.0,34.0,187.0,1.0,1.0,0.0,1.0,0.0,1135.0
1.0,25.0,105.0,1.0,0.0,1.0,1.0,0.0,1330.0
1.0,25.0,85.0,1.0,0.0,0.0,0.0,1.0,1474.0
1.0,27.0,150.0,1.0,0.0,0.0,0.0,0.0,1588.0
1.0,23.0,97.0,1.0,0.0,0.0,0.0,1.0,1588.0
```

- Excel
  - xlrd、xlwt、xlutils、openpyxl、xlsxwriter
  - pandas

list的一些有用操作

- range
  - range(start, end, step)
- enumerate
  - enumerate(list)
  - for index, val in enumerate(info):
    - print(index, val)
- zip
  - zip(list1, list2, list3, …)
  - namelist = [“Alpha”, “Bravo”, “Camp”]
  - agelist = [35, 46, 78]
  - heightlist = [1.78, 1.35, 1.65]
  - for (name, age, height) in zip(namelist, agelist, heightlist):
    - print(name, age, height)

生成式

- data1 = [0, 2, 4, 6, 8]
- 目标 data1每个数算平方放到data2
- 通常方案
  - data2 = []
  - for val in data1:
    - data2.append(val * val)

- 生成式方案
  - data2 = [val * val for val in data1]

- 生活如此美好~~~~~~~~~~~~
- 再加点料~~~~~~~~~~

- example = range(10)
- data2 = [val * val for val in example if val % 2 == 0]

生成式与生成器

- 举例
- data = [x * x for x in range(10000)]
- 内存占用较多
- gen_data = (x * x for x in range(10000))
  - print(next(gen_data))
- 这是一个生成器，不会占用过多的内存， 也可以用for循环
- for I in gen_data:
  - print(i)
- 更高级的使用： https://www.cnblogs.com/wj-1314/p/8490822.html

本节课小作业

- 练习课上的内容，做一做小练习
- note下载地址：http://staff.ustc.edu.cn/~zhyuey/python02.zip

大作业

- 与网络相关，使用API
- 把结果用适当方式展示出来 (GUI界面，网页）
- 上交内容：github/gitee库的公开地址，报告，小视频
- Deadline: 2019年3月31日23：59分
- 上交方式：发给两位助教，分配方式待定
- 参考API：
  - 空气质量
    - http://pm25.in/api_doc
  - 百度
    - http://ai.baidu.com/docs/#/
  - 聚合API
    - https://www.juhe.cn/docs
  - 地震
    - https://www.programmableweb.com/category/earthquakes/api