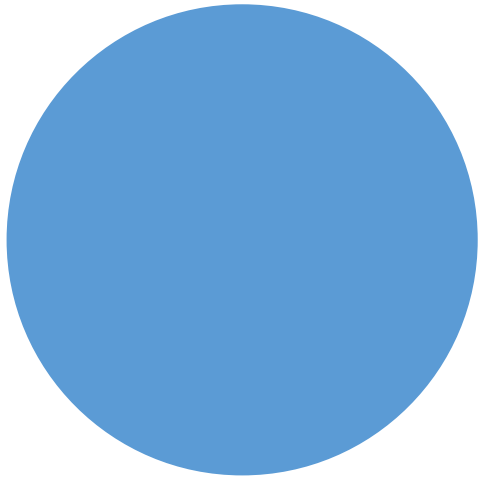
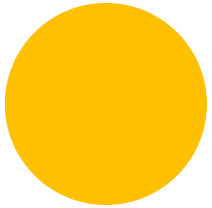



PYTHON

2019.11.14

- 解释型语言
  - 交互式编程
  - 移植性好
- 易读型语言
  - 易于阅读
  - 易于维护
  - 易于学习
- 胶水型语言
  - 扩展性好

移植性好 → 广泛的标准库

吉多·范罗苏姆(Guido van Rossum)
在1991年发布第一版Python
名字来源于电视喜剧《蒙提·派森的飞行马戏团》

Python

学习python要善用网络

https://docs.python.org/3/

https://www.runoob.com/python/python3-tutorial.html

搜索引擎

Python

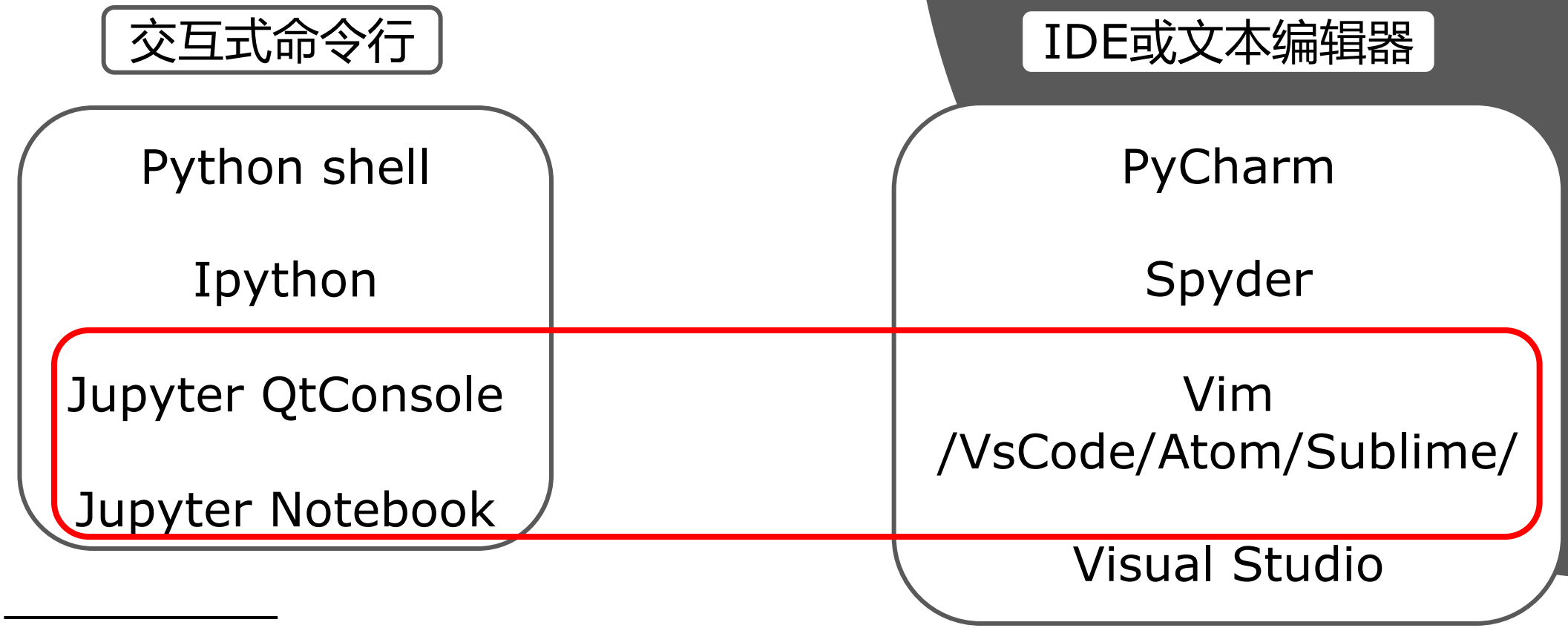
Python 安装

- 安装Anaconda（Python的一个发行版本，推荐）

或

- 在官网（https://www.python.org/）下载安装原版，然后用python自带的pip install命令安装需要的第三方库

编程环境


交互式命令行
IDE或文本编辑器
Python shell
Ipython
Jupyter QtConsole
Jupyter Notebook
PyCharm
Spyder
Vim
/VsCode/Atom/Sublime/
Visual Studio

编程环境-推荐搭建

windows linux图形界面

- 安装Anaconda
- 安装VS code

linux命令行

- 安装Anaconda
- 调试vim

Mac OS

- 安装Anaconda
- VS code ? PyCharm?

变量名

把这个字符串赋值给变量

```
message = “Hello Python World”
```

```
print(message)
```

打印变量

2.2.1 变量名的命名规则和注意事项

~~1_a~~

a_1

不需要声明变量类型，可以任意改变类型

print可以不用格式化输出（自动调用内建方法变成字符串后输出）

一行一个命令不用分号结束

变量

基本语法规则

标识名可以用字母、数字和下划线
标识名开头不能用数字
区分大小写
内建标识不可用于变量、函数

源码文件默认用UTF-8格式

单行注释以#开头
多行注释用’’’或”””在上下包围

使用缩进来区分代码块

```
a=0
b=0
for i in range(10):
    a+=1
b+=1
```

```
#!/usr/bin/python3
# 注释
# 注释
'''
注释
注释
'''
"""
注释
注释
"""
print ( “Hello, World!” )
```

对象

面向过程：数据+函数

面向对象：对象 包含 属性+方法

```
char *s="abc";
strlen(s);
```

```
name = "ada lovelace"

print(name.title())
```

属性是描述对象的数据
方法是对所在对象数据进行操作的函数

title()是一个实例对象（Instant Object）的方法（method）

```
object s{
    atri="abc"
    method strlen()
}
s.strlen()
```

面向对象编程方式可读性更好，修改需求方便，但是可控性差，程序结构复杂

Python是面向对象编程，但是它也可以采用面向过程形式

把大象放进冰箱

- 面向过程：

打开冰箱门（ ）
把大象放进冰箱（ ）
关上冰箱门（ ）

把狮子放进冰箱？
把大象放进冰箱并且温度调到-5度？

- 面向对象：

冰箱
冰箱.开门()
冰箱.关门()
冰箱.收纳一只动物(大象)
冰箱.放出一只动物(大象)
冰箱.调节温度(-5°)
冰箱.查找(大象)

大象
大象.吃()
大象.跑()
大象.进容器(冰箱)？

name = “ada lovelace” # name = ‘ada lovelace’

print(name.title())

print(“ada lovelace”.title())

nametitle = “ada lovelace”.title()

print(name.lower()) 把字符串都转化为小写

print(name.rstrip()) 删除字符串最后的回车

name.split()

字符串

查看说明文档
https://docs.python.org/3/

善用IDE补全和提示

学会看错误提示

Google / Baidu / Bing

“al” +”bc” =“albc”

“al” *4 =“alalalal”

大部分数字运算和C一样

3//2		3/2
10**6		pow(10，6)
str(15)	(str)(15)	(float)15
python		**C**

python默认都按照浮点数运算

乘方算符**，优先级最高

类型转换是内置函数

字符串，数字，类型转换

内置函数

		内置函数		
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()	

https://docs.python.org/zh-cn/3.7/library/functions.html

```
s = 123
t = "This is %d %s"%(1, s)

s = "apple"
s1 = "This is %d %s"
t = s1%(1, s)

"i am %(name)s, \
i am %(age)d years old, \
call me %(name)s"\
%{'age':26 , 'name':'jeck'}
```

字符串格式化：%方式

```
"%(name):-30.2f"%{'name':56.3}
```

```
In [153]: "%-30.2f"%(56.3)
Out[153]: '56.30                        '
```

```
"%-30.2f"%(56.3)
```

控制符 + - 0　格式符 与C相同

控制符

+ 右对齐:正数前加正号，负数前加负号；
- 左对齐:正数前无符号，负数前加负号；
空格 右对齐:正数前加空格，负数前加负号；
0 右对齐:正数前无符号，负数前加负号；用0填充空白处

格式符

c, o, x, d, e, E, f, F, g, G, %

s 获取传入对象的__str__方法的返回值，并将其格式化到指定位置

r 获取传入对象的__repr__方法的返回值，并将其格式化到指定位置

“My name is {}, my age is {}”.format(‘Jack’, 12)

“{0} of {1} blah {2} blah {0}”.format(‘a’, ‘b’, 7.2)

"--{name:*^10s}-- =={age:<10.2f}==".format(name='Jeck',age=26.457)

```
In [149]: "{0:-^30.2f}".format(56.3)
Out[149]: '------------56.30------------'
```

"{0:-^30.2f}".format(56.3)

指示位 填充符

< 靠左对齐
^ 居中
> 靠右对齐
= 靠右对齐，符号在最左侧

格式，与C相同

字符串格式化

print(*objects, sep=‘ ’, end=‘\n’, file=sys.stdout, flush=Flase)

```
s = “123”
a = 7
print(a, b)
print(a, sep=‘**’, end=‘; ’)
```

input(“提示语:”)　　读取一行，不包括回车符

```
input(“请输入名字：”)
```

输入输出

列表（list）

```
motor=['honda', 'Yamaha', 'Suzuki']
```

列表中的元素可以是任何类型的数据，可以混合

方括号包围表示列表结构

```
motor[2]='ducati'
#修改列表中的第3个元素
motor.append('ducati')
#在末尾添加一个元素
motor.insert(1, 'ducati')
#在列表的第1个元素位置插入新元素
bikesold = motor.pop(n)
#弹出第n个元素，没有参数则弹出最后一个
motor.remove('ducati')
#删除第一个符合的元素
del motor[0] #删除第一个元素
```

```
print(motor[n])
#打印列表的第个元素

print(motor[-n])
#打印列表倒数第n个元素
```

练习3.2, 3.3的内容

列表切片与复制

motor[start:end]

把列表的的从start开始到end前的元素取出，
返回的列表即原列表的切片

start 省略表示从列表开头开始切片

end 省略表示取到最后一个元素

start, end可以为负数，表示从末尾开始倒数，
必须为整数

```
a=[1,2,3,4,5,6,7,8]
b=a
c=a[:]
a[4]='I'
print(a)
print(b)
print(c)
```

切片是一个新列表

列表（list）II

```
motor=['honda', 'Yamaha', 'Suzuki']
```

```
len(motor)
#返回列表长度
motor + ['Toyota', 'Nissan']
#两个数组拼接
motor*4
#数组复制4次后拼接
'Fort' in motor
#判断'Fort'是否在motor列表中
motor.remove('ducati')
#删除第一个符合的元素
```

```
list.count(obj)
list.extend(seq)
list.sort()
list.reverse()
list.copy()
list.index(obj)
```

```
专用于数字列表
min(list)
max(list)
sum(list)
```

自行测试以上内容

元组就是不可变的列表

圆括号包围表示元组结构

```
dimensions=(200,50)
```

除了修改内容的操作，别的列表操作都可用于元组

```
dimensions[0]
len(dimensions)
for i in dimensions:
        print(i)
```

元组(Tuple)

字典(dictionary)

字典是采用关键字作为索引的数据列表

```
alien_0={'color':'green','points':5}
```

大括号包围，有键-值对，表示字典结构

键　　值

```
print(alien_0['color'])
```

字典中数值的使用与列表相同，只是把数字索引改为关键字索引

关键字也可以是数字

```
alien_0['color']='blue'
```

如果没有'color'这个键，则添加新的一个新的键-值对'color':'blue'进字典

如果有'color'这个键，则吧'color'的值改为'blue'

```
#删除键-值对
alien_0.pop('color')
del alien_0['points']
```

字典(dictionary)

```
alien_0={'color':'green', 'points':5}
```

```
alien_0.items()
\\列出所有键-值对
alien_0.keys()
\\列出所有键
alien_0.values()
\\列出所有值
```

字典没有排序的方法

~~alien_0.sort()~~

```
sorted(alien_0.keys())
list(alien_0.keys()).sort()
```

练习：在字典里找到某个值对应的键

集合是一个无序不重复元素集合

```
t ={'Huawei', 'Oppo', 'Xiaomi' }
s = {'Huawei', 'Lenovo', 'Acer'}
```

仅大括号包围
表示集合结构

```
t[0]
```

不可索引

```
for phone in t:
        print(phone)
```

可以遍历

集合(set)

```
t & s # t 和 s的并集
t | s  # t 和 s的并集
t - s  # 求差集（项在t中，但不在s中）
t ^ s  # 对称差集（项在t或s中，但不同时在两者中）
t <= s  # 测试t中的每个元素是否都在s中
t >= s  # 测试s中的每个元素是否都在t中
```

```
t.add('Vivo') # 集合中添加一项
t.update([1,2,3])  # 集合中添加多项
t.remove(1) #集合中删除一项
```

类型转换

列表	list()	list
元组	tuple()	a=tuple({'a':1,'b':2})
集合	set()	set
字典	dict()	a=dict([['a',1],['b',2]])

将字典变为列表/元组/集合时，只取键值

变为字典时，输入必须时二维列表/元组
键值重复时只取最后一个

四种类型可以相互嵌套

```
magicians = [‘alice’, ‘David’, ‘Carolina’]
for magician in magicians:
//一次读取列表magicians中的每个成员
//每次读出时把成员存入变量magician中
        print(magician)
        print(“I am”+magician)
        //循环中的操作，打印magician
```

Python中的循环“计数”用的是列表

用缩进来表示层级 4.2节内容

控制语句 for循环

range函数

range(start, stop, [step])生成一个等差数列

数列的第一个数字是start

相邻数字间的差值为step，
step>0递增，step<0递减，省略step时step=1

数列增长到stop前一个数字停止

start, stop, step都必须是整数

```
for i in range(0,10):
        print(i)
```

```
for(int i=0;i<10;i++):
        printf("%d\n",i);
```

```
l= [i**2 for i in range(1,11)]
\\常用的构建有一定规律数字列表的方法
```

range函数的返回值是一个迭代器，但是可以用类型转换变成列表

```
while 判断条件：
        满足条件时运行的语句
```

要注意的点都和C语言一样

continue

break

```
prompt="\rTell me sth, I will repeat:"
prompt+="\n(Enter 'quit' to end the program)\n"
message = ""
while message !='quit' :
        print(message)
        message = input(prompt)
```

while循环

```
if 判断条件：
        满足条件时运行的语句
elif 条件：
        运行语句
else：
        运行语句
```

C 中 else if

用缩进来表示层级

条件判断中与C不同的地方

- 可以直接对字符串做<,>,=,!= >=,<=操作
- && => and,
- || => or
- ! => not
- 关键字in, 列表内是否存在某个元素/not in
- 有专门的布尔值True和False来表示判断结果

if 及条件判断

函数

定义函数
的关键字

具有默认
值的参数

```
def func(para1, para2, para3="default") :
    print(para1)
    print(para2)
    print(para3)
    return para1+para2, para1+para3
```

通过缩进判断哪些语
句是属于一个函数

可以返回多个值
实际是打包成一个元组返回

```
func("a", "b")
func(para1="a", para2="b")

func("a", "b", "c")
func(para3="c", para2="b", para1="a")
```

mutable(可变）数据类型：
列表，字典

指针传递

immutable(不可变）数据类型：
整数，浮点数，元组

值传递

函数参数传递

```
def func(para1, *para2) :
        print(para1)
        print(para2)
```

```
func("a", "b", "c", "d")
```

```
def func(para1, **para_dict) :
        print(para1)
        print(para_dict)
```

```
func("a", para2="b", para3="c", para4="d")
```

函数中的可变参数

```
class Dog() :
        """定义了一个类Dog"""
    sound=["wang","wu"] #类变量
    tail=1
    def __init__(self, para1, para2):
        """实例初始化"""
        self.name=para1 #实例变量
        self.age=para2
    def sit(self):
        """一个方法，显示小狗坐下"""
        print(self.name+"is sitting")
        self.__tickle()
    def __tickle(self):#私有方法
        print('tickling…')
```

```
mydog=Dog("will",6)
\\创建了一个实例（对象）
```

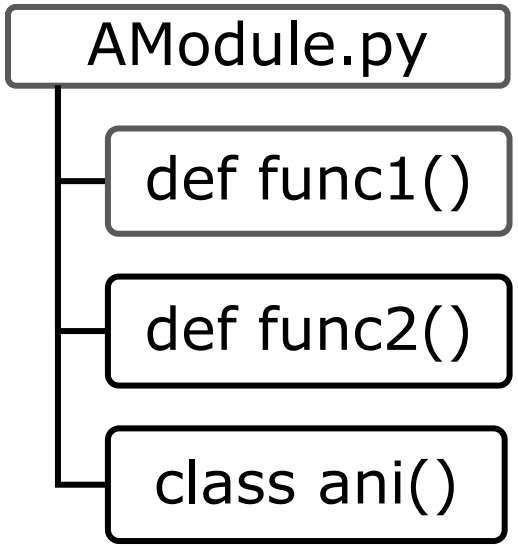
```
mydog.name
\\调用了mydog这个对象里的name属性
mydog.sit()
\\调用了对象的方法
```

```
class Puppy(Dog) :
    def __init__(self, para1, para2):
        Dog.__init__(self, para1, para2)
        self.face = 'cute'
```

类

第9章

模块



一个.py文件，既可以当做独立的脚本运行，又可以被其他的程序导入来使用其函数和类定义

```
import Amodule

Amodule.func1()
Amodule.func1()
```

```
import Amodule.func1 as myfunc

myfunc()
```

```
from Amodule import func1 [as myfunc]

func1()
```

```
from Amodule import *

func1()
func2()
```

```
fp=open("text.txt","r")
for line in fp:
    print(line)
```

"r" "w" "r+" "w+" "a" "b"

```
with open("text.txt","r") as fp:
    for line in fp:
        print(line)
```

```
fp.read(n)
fp.seek(n, [whence=0])


fp.write("写入的字符串")

fp.readline(n)
fp.readlines(n)

fp.close()
```

二进制：n个字节
asic: n个字符

文件

模式	r	r+	w	w+	a	a+
读	+	+		+		+
写		+	+	+	+	+
创建			+	+	+	+
覆盖			+	+		
指针在开始	+	+	+	+		
指针在结尾					+	+

```
fp=open(filename,”rb”)
data = fp.read(4)
```

```
data
b'5\x9d\x82\xc3
'
```

```
import struct
…
data_float = struct.unpack(“f”, data)[0]
```

二进制数据

Format	c Type	Python	Note
x	pad byte	no value	
c	char	string of length 1	
b	signedchar	integer	
B	unsignedchar	integer	
?	_Bool	bool	(1)
h	short	integer	
H	unsignedshort	integer	
i	int	integer	
I	unsignedint	integer or long	
l	long	integer	
L	unsignedlong	long	
q	longlong	long	(2)
Q	unsignedlonglong	long	(2)
f	float	float	
d	double	float	
s	char[]	string	
p	char[]	string	
P	void*	long	

```
import pickle
```

```
data1 = {'a': [1, 2.0, 3, 4+6j],
         'b': ('string', u'Unicode string'),
         'c': None}
```

```
fp=open(filename,”wb”)
pickle.dump(data1, fp)
```

```
fp=open(filename,”rb”)
data1=pickle.load(fp)
```

二进制数据

```
try：
        运行的语句
except 错误类型：
        出现错误时的对应操作
else：
        try语句块运行成功后对应操作
```

异常处理

课后练习

1. 用python写一个词频统计，要求读取一个文本文件，记录里面的所有单词并计数。然后输出词频统计到另一个文件
2. 写一个学生数据记录程序：
   1. 运行软件后，读取本地数据库文件，不存在数据库文件则创建一个。
   2. 提示用户输入“学生姓名 年龄 性别 班级 学号”的信息，程序读取信息后存入文件，并等待下一个输入或结束指令
3. 写一个学生数据整理程序：
   1. 运行软件后读取本地数据库文件，如果不存在给出提示。
   2. 显示学生信息列表
   3. 统计男生/女生人数，统计班级人数
   4. 把学生信息按照姓名顺序排序并重新保存(.sort(key=func))

选做：遍历一个文件夹中的所有文件和子文件夹，打印出其中所有.txt文件的路径。（自行学习os.path和sys模块）