Lab2.5 Python部分基础知识介绍与类库基础

Python基础

Python解释器

CPython
- 官方版本的解释器。在命令行下运行python就是启动CPython解释器
- CPython是用C语言开发的，所以叫CPython
- CPython是使用最广的Python解释器

IPython
- IPython是基于CPython之上的一个交互式解释器，IPython只是在交互方式上有所增强，但是执行Python代码的功能和CPython是完全一样的。
- CPython用>>>作为提示符，而IPython用In [序号]:作为提示符。

Python基础

标识符

- 标识符：用于变量、函数、类、模块等的名称。

- 标识符有如下特定的规则:

① 区分大小写。如: a和A是不同的
② 第一个字符必须是字母、下划线。其后的字符是:字母、数字、下划线
③ 不能使用关键字。比如: if, or, while等
④ 以双下划线开头和结尾的名称通常有特殊含义,尽量避免这种写法。
如:__init__是类的构造函数。

Python基础

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Python基础

对 象

- Python中一切皆对象。

- 每个对象由:标识(identity)、类型(type)、 value (值组成） 。
  ① 标识用于唯一标识对象,通常对应于对象在计算机内存中的地址。使用内置函数id(obj)可返回对象obj的标识。
  ② 类型用于表示对象存储的"数据"的类型。类型可以限制对象的取值范围以及可执行的操作。可使用type(obj)获得对象的所属类型。
  ③ 值表示对象所存储的数据的信息。使用print(obj)可以直接打印出值。

对象的本质: 一个内存块,拥有特定的值,支持特定类型的相关操作。

Python基础

引 用

- 在Python中，变量也称为：对象的引用
- 变量存储的就是对象的地址
- 变量位于：栈内存（压栈出栈)
- 对象位于：堆内存

– Python是动态类型语言，变量不需要显式声明类型。根据变量引用的对象，Python解释器自动确定数据类型。

– Python是强类型语言，每个对象都有数据类型，只支持该类型支持的操作。

Python基础

a=10, b=21

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 31
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -11
* Multiplication	Multiplies values on either side of the operator	a * b = 210
/ Division	Divides left hand operand by right hand operand	b / a = 2.1
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 1
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity):	9//2 = 4 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0

Python基础

a=10, b=21

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	(a!= b) is true.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

Python基础

a=10, b=21

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

Python基础

逻辑运算符	含义	基本格式	说明
and	逻辑与运算，等价于数学中的“且”	a and b	当 a 和 b 两个表达式都为真时，a and b 的结果才为真，否则为假。
or	逻辑或运算，等价于数学中的“或”	a or b	当 a 和 b 两个表达式都为假时，a or b 的结果才是假，否则为真。
not	逻辑非运算，等价于数学中的“非”	not a	如果 a 为真，那么 not a 的结果为假；如果 a 为假，那么 not a 的结果为真。相当于对 a 取反。

Python基础

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here **is** results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here **is not** results in 1 if id(x) is not equal to id(y).

Python基础

数据类型：

Number（整数、浮点数、复数、布尔值）
String（字符串）
tuple（元组）
list（列表）
dict（字典）
set（集合）

python2	python3	备注
x = 1000000000000L	x = 1000000000000	python2中的十进制长整型在python3中被替换为十进制普通整数
x = 0xFFFFFFFFFFFFL	x = 0xFFFFFFFFFFFF	python2里的十六进制长整型在python3里被替换为十六进制的普通整数
long(x)	int(x)	python3没有long()
type(x) is long	type(x) is int	python3用int判断是否为整型
isinstance(x, long)	isinstance(x, int)	int检查整数类型

Python基础

String
[:] + *

str = 'Hello World!'

- print (str) # Prints complete string
- print (str[0]) # Prints first character of the string
- print (str[2:5]) # Prints characters starting from 3rd to 5th
- print (str[2:]) # Prints string starting from 3rd character
- print (str * 2) # Prints string two times
- print (str + "TEST") # Prints concatenated string

Python基础

List
[:] + *

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']
```

- print (list) # Prints complete list
- print (list[0]) # Prints first element of the list
- print (list[1:3]) # Prints elements starting from 2nd till 3rd
- print (list[2:]) # Prints elements starting from 3rd element
- print (tinylist * 2) # Prints list two times
- print (list + tinylist) # Prints concatenated lists

Python基础

List

Methods :

- list.append(obj)
- list.count(obj)
- list.extend(seq)
- list.index(obj)
- list.insert(index, obj)
- list.pop(obj = list[-1])
- list.remove(obj)
- list.reverse()
- list.sort([func])

Python基础

Tuple
[:] + *

```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )
tiny_tuple = (123, 'john')
```

- print (tuple) # Prints complete tuple
- print (tuple[0]) # Prints first element of the tuple
- print (tuple[1:3]) # Prints elements starting from 2nd till 3rd
- print (tuple[2:]) # Prints elements starting from 3rd element
- print (tiny_tuple * 2) # Prints tuple two times
- print (tuple + tiny_tuple) # Prints concatenated tuple

Python基础

Dictionary

```
dict = {}
dict['one'] = "This is one"
dict[2] = "This is two  “
tinydict = {'name': 'john','code':6734, 'dept': 'sales'}
```

- print (dict['one']) # Prints value for 'one' key
- print (dict[2]) # Prints value for 2 key
- print (tinydict) # Prints complete dictionary
- print (tinydict.keys()) # Prints all the keys
- print (tinydict.values()) # Prints all the values

Python基础

Dictionary
Method & Description :

- dict.clear()

Removes all elements of dictionary dict

- dict.copy()

Returns a shallow copy of dictionary dict

- dict.fromkeys()

Create a new dictionary with keys from seq and values set to value.

- dict.get(key, default=None)

For key key, returns value or default if key not in dictionary

- dict.has_key(key)

Removed, use the in operation instead

Python基础

Dictionary
Method & Description :

- dict.items()

Returns a list of dict's (key, value) tuple pairs

- dict.keys()

Returns list of dictionary dict's keys

- dict.setdefault(key, default = None)

Similar to get(), but will set dict[key] = default if key is not already in dict

- dict.update(dict2)

Adds dictionary dict2's key-values pairs to dict

- dict.values()

Returns list of dictionary dict's values

Python基础

条件判断

```
age = 3
if age >= 18:
    print('adult')
elif age >= 6:
    print('teenager')
else:
    print('kid')
```

Python基础

循环

for 循环

```
names = ['Michael', 'Bob', 'Tracy']
for name in names:
    print(name)
```

while循环

```
n = 1
while n <= 100:
    print(n)
    n = n + 1
print('END')
```

break语句可以在循环过程中直接退出循环
continue语句可以提前结束本轮循环，并直接开始下一轮循环。

Python基础

函数

- 定义函数时，需要确定函数名和参数个数
- 如果有必要，可以先对参数的数据类型做检查
- 函数体内部可以用return随时返回函数结果
- 函数执行完毕也没有return语句时，自动return None
- 函数可以同时返回多个值，但其实就是一个tuple

```
def my_abs(x):
    if x >= 0:
        return x
    else:
        return -x
```

Python基础

函数的参数传递本质上就是：从实参到形参的赋值操作。 Python 中“一切皆对象”，所有的赋值操作都是“引用的赋值”。所以，Python 中参数的传递都是“引用传递”，不是“值传递”。具体操作时分为两类：

1. 对“可变对象”进行“写操作”，直接作用于原对象本身。

2. 对“不可变对象”进行“写操作”，会产生一个新的“对象空间”，并用新的值填充这块空间。（起到其他语言的“值传递”效果，但不是“值传递”）

可变对象有：

字典、列表、集合、自定义的对象等

不可变对象有：

数字、字符串、元组、function 等

Python基础

面向对象

- 类(Class): 用来描述具有相同的属性和方法的对象的集合。它定义了该集合中每个对象所共有的属性和方法。对象是类的实例。

- 类变量：类变量在整个实例化的对象中是公用的。类变量定义在类中且在函数体之外。类变量通常不作为实例变量使用。

- 数据成员：类变量或者实例变量, 用于处理类及其实例对象的相关的数据。

- 方法重写：如果从父类继承的方法不能满足子类的需求，可以对其进行改写，这个过程叫方法的覆盖（override），也称为方法的重写。

Python基础

- 局部变量：定义在方法中的变量，只作用于当前实例的类。

- 实例变量：在类的声明中，属性是用变量来表示的。这种变量就称为实例变量，是在类声明的内部但是在类的其他成员方法之外声明的。

- 继承：即一个派生类（derived class）继承基类（base class）的字段和方法。继承也允许把一个派生类的对象作为一个基类对象对待。

- 实例化：创建一个类的实例，类的具体对象。

- 方法：类中定义的函数。

- 对象：通过类定义的数据结构实例。对象包括两个数据成员（类变量和实例变量）和方法。

Python基础

如何定义类

class <类名>(父类):

通常，如果没有合适的继承类，就使用object类，这是所有类最终都会继承的类。

```
class Student(object):
    pass
```

类的初始化

__init__ 是初始化方法，用于实例的初始化。

__init__方法的第一个参数永远是self，表示创建的实例本身，因此，在__init__方法内部，就可以把各种属性绑定到self，因为self就指向创建的实例本身。

```
class Student(object):

    def __init__(self, name, score):
        self.name = name
        self.score = score
```

有了__init__方法，在创建实例的时候，就不能传入空的参数了，必须传入与__init__方法匹配的参数，但self不需要传，Python解释器自己会把实例变量传进去。

Python基础

类属性和实例属性

- 实例能访问其类属性
- 类无法访问实例属性

实例属性每个实例各自拥有，互相独立
类属性有且只有一份。

```
class Student:
    unit = 'nankai'

    def __init__(self, name, number):
        self.name = name
        self.number = number
```

```
john = Student('John',1234)
print(john.name, john.number, john.unit)
```

```
John 1234 nankai
```

```
Student.unit
```

```
'nankai'
```

```
Student.name
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_89908\3290574333.py in <module>
----> 1 Student.name

AttributeError: type object 'Student' has no attribute 'name'
```

Python基础

类中的方法

在类的内部，使用 def 关键字来定义方法，与一般函数定义不同，类方法必须第一个参数为 self，self 代表的是类的实例（即你还未创建类的实例），其他参数和普通函数是完全一样。

```
class Student:
    unit = 'nankai'

    def __init__(self, name, number):
        self.name = name
        self.number = number

    def study(self, major):
        print(f"{self.name} studies {major}")
```

```
john = Student('John',1234)
john.study('law')
```

```
John studies law
```

类库基础

本课程使用到的第三方类库基础介绍请参见单独的ipynb文件演示。

包活numpy, pandas, scipy, matplotlib。

在学习后请完成习题，巩固知识。