AI 基础：Python 简易入门

原创：机器学习初学者 机器学习初学者 1周前

0.导语

Python是一种跨平台的计算机程序设计语言。是一种面向对象的动态类型语言，最初被设计用于编写自动化脚本(shell)，随着版本的不断更新和语言新功能的添加，越来越多被用于独立的、大型项目的开发。

在此之前，我已经写了以下几篇AI基础的快速入门，本篇文章讲解**python语言的基础部分，也是后续内容的基础。**

已发布：

AI 基础：Numpy 简易入门

AI 基础：Pandas 简易入门

AI 基础：Scipy(科学计算库) 简易入门

AI基础：数据可视化简易入门（matplotlib和seaborn）

后续持续更新

本文代码可以在**github**下载：

https://github.com/fengdu78/Data-Science-Notes/tree/master/1.python-basic

文件名：**Python_Basic.ipynb**

1 Python数据类型

1.1 字符串

在Python中用引号引起来的字符集称之为字符串，比如：'hello'、"my Python"、"2+3"等都是字符串 Python中字符串中使用的引号可以是单引号、双引号跟三引号

```
print ('hello world!')
```

```
hello world!
```

```
c = 'It is a "dog"!'
print (c)
```

```
It is a "dog"!
```

```
c1= "It's a dog!"
print (c1)
```

```
It's a dog!
```

```
c2 = """hello
world
!"""
print (c2)
```

```
hello
world
!
```

- 转义字符''

转义字符\可以转义很多字符，比如\n表示换行，\t表示制表符，字符\本身也要转义，所以\\表示的字符就是\

```
print ('It\'s a dog!')
print ("hello world!\nhello Python!")
print ('\\\t\\')
```

```
It's a dog!
hello world!
hello Python!
\       \
```

原样输出引号内字符串可以使用在引号前加r

```
print (r'\\\t\\')
```

```
\\\t\\
```

- 子字符串及运算

```
s = 'Python'
print( 'Py' in s)
print( 'py' in s)
```

```
True
False
```

取子字符串有两种方法，使用[]索引或者切片运算法[:]，这两个方法使用面非常广

```
print (s[2])
```

```
t
```

```
print (s[1:4])
```

```
yth
```

- 字符串连接与格式化输出

```
word1 = '"hello"'
word2 = '"world"'
sentence = word1.strip('"') + ' ' + word2.strip('"') + '!'

print( 'The first word is %s, and the second word is %s' %(word1, word2))
print (sentence)
```

```
The first word is "hello", and the second word is "world"
hello world!
```

1.2 整数与浮点数

整数

Python可以处理任意大小的整数，当然包括负整数，在程序中的表示方法和数学上的写法一模一样

```
i = 7
print (i)
```

```
7
```

```
7 + 3
```

```
10
```

```
7 - 3
```

```
4
```

```
7 * 3
```

```
21
```

```
7 ** 3
```

```
343
```

```
7 / 3#Python3之后，整数除法和浮点数除法已经没有差异
```

```
2.3333333333333335
```

```
7 % 3
```

```
1
```

```
7//3
```

```
2
```

浮点数

```
7.0 / 3
```

```
2.3333333333333335
```

```
3.14 * 10 ** 2
```

```
314.0
```

其它表示方法

```
0b1111
```

```
15
```

```
0xff
```

```
255
```

```
1.2e-5
```

```
1.2e-05
```

更多运算

```
import math

print (math.log(math.e)) # 更多运算可查阅文档
```

```
1.0
```

1.3 布尔值

```
True
```

```
True
```

```
False
```

```
False
```

```
True and False
```

```
False
```

```
True or False
```

```
True
```

```
not True
```

```
False
```

```
True + False
```

```
1
```

```
18 >= 6 * 3 or 'py' in 'Python'
```

```
True
```

```
18 >= 6 * 3 and 'py' in 'Python'
```

```
False
```

```
18 >= 6 * 3 and 'Py' in 'Python'
```

```
True
```

1.4 日期时间

```
import time

now = time.strptime('2016-07-20', '%Y-%m-%d')
print (now)
```

```
time.struct_time(tm_year=2016, tm_mon=7, tm_mday=20, tm_hour=0, tm_min=0, tm_sec=0, tm_wday=2, tm_yday=
```

```
time.strftime('%Y-%m-%d', now)
```

```
'2016-07-20'
```

```
import datetime

someDay = datetime.date(1999,2,10)
anotherDay = datetime.date(1999,2,15)
```

```
deltaDay = anotherDay - someDay
deltaDay.days
```

5

还有其他一些datetime格式

表示	含义
%a	星期的简写。如 星期三为Web
%A	星期的全写。如 星期三为Wednesday
%b	月份的简写。如4月份为Apr
%B	月份的全写。如4月份为April
%c	日期时间的字符串表示。（如： 04/07/10 10:43:39）
%d	日在这个月中的天数（是这个月的第几天）
%f	微秒（范围[0,999999]）
%H	小时（24小时制，[0, 23]）
%I	小时（12小时制，[0, 11]）
%j	日在年中的天数 [001,366]（是当年的第几天）
%m	月份（[01,12]）
%M	分钟（[00,59]）
%p	AM或者PM
%S	秒（范围为[00,61]，为什么不是[00, 59]，参考python手册~_~）
%U	周在当年的周数当年的第几周），星期天作为周的第一天
%w	今天在这周的天数，范围为[0, 6]，6表示星期天
%W	周在当年的周数（是当年的第几周），星期一作为周的第一天
%x	日期字符串（如：04/07/10）
%X	时间字符串（如：10:43:39）
%y	2个数字表示的年份
%Y	4个数字表示的年份
%z	与utc时间的间隔 （如果是本地时间，返回空字符串）
%Z	时区名称（如果是本地时间，返回空字符串）
%%	%% => %

- 查看变量类型

```
type(None)
```

NoneType

```

```

```
type(1.0)
```

```
float
```

```
type(True)
```

```
bool
```

```
s="NoneType"
type(s)
```

```
str
```

- 类型转换

```
str(10086)
```

```
'10086'
```

```
?float()
```

```
float(10086)
```

```
10086.0
```

```
int('10086')
```

```
10086
```

```
complex(10086)
```

```
(10086+0j)
```

2 Python数据结构

列表（list）、元组（tuple）、集合（set）、字典（dict）

2.1 列表(list)

用来存储一连串元素的容器，列表用[]来表示，其中元素的类型可不相同。

```
mylist= [0, 1, 2, 3, 4, 5]
print (mylist)
```

```
[0, 1, 2, 3, 4, 5]
```

列表索引和切片

```
# 索引从0开始，含左不含右
print ('[4]=', mylist[4])
print ('[-4]=', mylist[-4])
print ('[0:4]=', mylist[0:4])
print ('[:4]=', mylist[:4])#dddd
print( '[4:]=', mylist[4:])
print ('[0:4:2]=', mylist[0:4:2])
print ('[-5:-1:]=', mylist[-5:-1:])
print ('[-2::-1]=', mylist[-2::-1])
```

```
[4]= 4
[-4]= 2
[0:4]= [0, 1, 2, 3]
[:4]= [0, 1, 2, 3]
[4:]= [4, 5]
[0:4:2]= [0, 2]
[-5:-1:]= [1, 2, 3, 4]
[-2::-1]= [4, 3, 2, 1, 0]
```

修改列表

```
mylist[3] = "小月"
print (mylist[3])

mylist[5]="小楠"
print (mylist[5])

mylist[5]=19978
print (mylist[5])
```

```
小月
小楠
19978
```

```
print (mylist)
```

```
[0, 1, 2, '小月', 4, 19978]
```

插入元素

```
mylist.append('han') # 添加到尾部
mylist.extend(['long', 'wan'])
print (mylist)
```

```
[0, 1, 2, '小月', 4, 19978, 'han', 'long', 'wan']
```

```
scores = [90, 80, 75, 66]
mylist.insert(1, scores) # 添加到指定位置
mylist
```

```
[0, [90, 80, 75, 66], 1, 2, '小月', 4, 19978, 'han', 'long', 'wan']
```

```
a=[]
```

删除元素

```
print (mylist.pop(1)) # 该函数返回被弹出的元素，不传入参数则删除最后一个元素
print (mylist)
```

```
[90, 80, 75, 66]
[0, 1, 2, '小月', 4, 19978, 'han', 'long', 'wan']
```

判断元素是否在列表中等

```
print( 'wan' in mylist)
print ('han' not in mylist)
```

```
True
False
```

```
mylist.count('wan')
```

```
1
```

```
mylist.index('wan')
```

```
8
```

range函数生成整数列表

```
print (range(10))
print (range(-5, 5))
print (range(-10, 10, 2))
print (range(16, 10, -1))
```

```
range(0, 10)
range(-5, 5)
range(-10, 10, 2)
range(16, 10, -1)
```

2.2 元组(tuple)

元组类似列表，元组里面的元素也是进行索引计算。列表里面的元素的值可以修改，而元组里面的元素的值不能修改，只能读取。元组的符号是()。

```
studentsTuple = ("ming", "jun", "qiang", "wu", scores)
studentsTuple
```

```
('ming', 'jun', 'qiang', 'wu', [90, 80, 75, 66])
```

```
try:
    studentsTuple[1] = 'fu'
except TypeError:
    print ('TypeError')
```

```
TypeError
```

```
scores[1]= 100
studentsTuple
```

```
('ming', 'jun', 'qiang', 'wu', [90, 100, 75, 66])
```

```
'ming' in studentsTuple
```

```
True
```

```
studentsTuple[0:4]
```

```
('ming', 'jun', 'qiang', 'wu')
```

```
studentsTuple.count('ming')
```

```
1
```

```
studentsTuple.index('jun')
```

```
1
```

```
len(studentsTuple)
```

```
5
```

2.3 集合(set)

Python中集合主要有两个功能，一个功能是进行集合操作，另一个功能是消除重复元素。集合的格式是：set()，其中()内可以是列表、字典或字符串，因为字符串是以列表的形式存储的

```
studentsSet = set(mylist)
print (studentsSet)
```

```
{0, 1, 2, 'han', 4, '小月', 19978, 'wan', 'long'}
```

```
studentsSet.add('xu')
print (studentsSet)
```

```
{0, 1, 2, 'han', 4, '小月', 19978, 'wan', 'long', 'xu'}
```

```
studentsSet.remove('xu')
print (studentsSet)
```

```
{0, 1, 2, 'han', 4, '小月', 19978, 'wan', 'long'}
```

```
mylist.sort()#会出错
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-69-7309caa8a1d6> in <module>()
----> 1 mylist.sort()


TypeError: '<' not supported between instances of 'str' and 'int'
```

```
a = set("abcnmaaaaggsng")
print ('a=', a)
```

```
a= {'b', 'a', 'm', 'c', 'g', 's', 'n'}
```

```
b = set("cdfm")
print ('b=', b)
```

```
b= {'m', 'd', 'c', 'f'}
```

```
#交集
x = a & b
print( 'x=', x)
```

```
x= {'m', 'c'}
```

```
#并集
y = a | b
print ('y=', y)
#差集
z = a - b
print( 'z=', z)
#去除重复元素
new = set(a)
print( z)
```

```
y= {'b', 'a', 'f', 'd', 'm', 'c', 'g', 's', 'n'}
z= {'b', 'a', 'g', 's', 'n'}
{'b', 'a', 'g', 's', 'n'}
```

2.4字典(dict)

Python中的字典dict也叫做关联数组，用大括号{}括起来，在其他语言中也称为map，使用键-值（key-value）存储，具有极快的查找速度，其中key不能重复。

```
k = {"name":"weiwei", "home":"guilin"}
print (k["home"])
```

```
guilin
```

```
print( k.keys())
print( k.values())
```

```
dict_keys(['name', 'home'])
dict_values(['weiwei', 'guilin'])
```

添加、修改字典里面的项目

```
k["like"] = "music"
k['name'] = 'guangzhou'
print (k)
```

```
{'name': 'guangzhou', 'home': 'guilin', 'like': 'music'}
```

```
k.get('edu', -1) # 通过dict提供的get方法，如果key不存在，可以返回None，或者自己指定的value
```

```
-1
```

删除key-value元素

```
k.pop('like')
print (k)
```

```
{'name': 'guangzhou', 'home': 'guilin'}
```

2.5 列表、元组、集合、字典的互相转换

```
type(mylist)
```

```
list
```

```
tuple(mylist)
```

```
(0, 1, 2, '小月', 4, 19978, 'han', 'long', 'wan')
```

```
list(k)
```

```
['name', 'home']
```

```
zl = zip(('A', 'B', 'C'), [1, 2, 3, 4]) # zip可以将列表、元组、集合、字典‘缝合’起来
print (zl)
print (dict(zl))
```

```
<zip object at 0x0000015AFAA612C8>
```

```
{'A': 1, 'B': 2, 'C': 3}
```

3 Python控制流

在Python中通常的情况下程序的执行是从上往下执行的，而某些时候我们为了改变程序的执行顺序，使用控制流语句控制程序执行方式。Python中有三种控制流类型：顺序结构、分支结构、循环结构。

另外，Python可以使用分号";"分隔语句，但一般是使用换行来分隔；语句块不用大括号"{}"，而使用缩进（可以使用四个空格）来表示

3.1 顺序结构

```
s = '7'
num = int(s) # 一般不使用这种分隔方式
num -= 1 # num = num - 1
num *= 6 # num = num * 6
print (num)
```

```
36
```

3.2 分支结构：Python中if语句是用来判断选择执行哪个语句块的

if <True or Flase表达式>:

    执行语句块

elif <True or Flase表达式>：

    执行语句块

else：　　# 都不满足

    执行语句块

#elif子句可以有多条，elif和else部分可省略

```
salary = 1000
if salary > 10000:
```

```
    print ("Wow!!!!!!!")
elif salary > 5000:
    print ("That's OK.")
elif salary > 3000:
    print ("5555555555")
else:
    print (".........")
```

....

3.3 循环结构

while 循环

while <True or Flase表达式>:

循环执行语句块

else：　　# 不满足条件

执行语句块

#else部分可以省略

```
a = 1
while a < 10:
    if a <= 5:
        print (a)
    else:
        print ("Hello")
    a = a + 1
else:
    print ("Done")
```

```
1
2
3
4
5
Hello
Hello
Hello
Hello
Done
```

- for 循环 for (条件变量) in (集合)：

  执行语句块

“集合”并不单指set，而是“形似”集合的列表、元组、字典、数组都可以进行循环

条件变量可以有多个

```
heights = {'Yao':226, 'Sharq':216, 'AI':183}
for i in heights:
    print (i, heights[i])
```

```
Yao 226
Sharq 216
AI 183
```

```
for key, value in heights.items():
    print(key, value)
```

```
Yao 226
Sharq 216
AI 183
```

```
total = 0
for i in range(1, 101):
    total += i#total=total+i
print (total)
```

```
5050
```

3.4 break、continue和pass

break:跳出循环

continue:跳出当前循环,继续下一次循环

pass:占位符，什么也不做

```
for i in range(1, 5):
    if i == 3:
        break
    print (i)
```

```
1
2
```

```
for i in range(1, 5):
    if i == 3:
        continue
    print (i)
```

```
1
2
4
```

```
for i in range(1, 5):
    if i == 3:
        pass
    print (i)
```

```
1
2
3
4
```

3.5 列表生成式

三种形式

- [<表达式> for (条件变量) in (集合)]
- [<表达式> for (条件变量) in (集合) if <'True or False'表达式>]
- [<表达式> if <'True or False'表达式> else <表达式> for (条件变量) in (集合)]

```
fruits = ['"Apple', 'Watermelon', '"Banana"']
[x.strip('"') for x in fruits]
```

```
['Apple', 'Watermelon', 'Banana']
```

```
# 另一种写法
test_list=[]
for x in fruits:
    x=x.strip('"')
    test_list.append(x)
test_list
```

```
['Apple', 'Watermelon', 'Banana']
```

```
[x ** 2 for x in range(21) if x%2]
```

```
[1, 9, 25, 49, 81, 121, 169, 225, 289, 361]
```

```
# 另一种写法
test_list=[]
for x in range(21):
    if x%2:
        x=x**2
        test_list.append(x)
test_list
```

```
[1, 9, 25, 49, 81, 121, 169, 225, 289, 361]
```

```
[m + n for m in 'ABC' for n in 'XYZ']
```

```
['AX', 'AY', 'AZ', 'BX', 'BY', 'BZ', 'CX', 'CY', 'CZ']
```

```
# 另一种写法
test_list=[]
for m in 'ABC':
    for n in 'XYZ':
        x=m+n
        test_list.append(x)
test_list
```

```
['AX', 'AY', 'AZ', 'BX', 'BY', 'BZ', 'CX', 'CY', 'CZ']
```

```
d = {'x': 'A', 'y': 'B', 'z': 'C' }
[k + '=' + v for k, v in d.items()]
```

```
['x=A', 'y=B', 'z=C']
```

```
# 另一种写法
test_list=[]
for k, v in d.items():
    x=k + '=' + v
    test_list.append(x)
test_list
```

```
['x=A', 'y=B', 'z=C']
```

4 Python函数

函数是用来封装特定功能的实体，可对不同类型和结构的数据进行操作，达到预定目标。

4.1 调用函数

- Python内置了很多有用的函数，我们可以直接调用，进行数据分析时多数情况下是通过调用定义好的函数来操作数据的

```
str1 = "as"
int1 = -9
print (len(str1))
print (abs(int1))
```

```
2
9
```

```
fruits = ['Apple', 'Banana', 'Melon']
fruits.append('Grape')
print (fruits)
```

```
['Apple', 'Banana', 'Melon', 'Grape']
```

4.2 定义函数

当系统自带函数不足以完成指定的功能时，需要用户自定义函数来完成。def 函数名()：函数内容 函数内容 <return 返回值>

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

my_abs(-9)
```

```
9
```

可以没有return

```
def filter_fruit(someList, d):
    for i in someList:
        if i == d:
            someList.remove(i)
        else:
            pass

print (filter_fruit(fruits, 'Melon'))
print (fruits)
```

```
None
['Apple', 'Banana', 'Grape']
```

多个返回值的情况

```
def test(i, j):
    k = i * j
    return i, j, k

a , b , c = test(4, 5)
print (a, b , c)
type(test(4, 5))
```

```
4 5 20
tuple
```

4.3 高阶函数

- 把另一个函数作为参数传入一个函数，这样的函数称为高阶函数

函数本身也可以赋值给变量，函数与其它对象具有同等地位

```
myFunction = abs
myFunction(-9)
```

```
9
```

- 参数传入函数

```
def add(x, y, f):
    return f(x) + f(y)
```

```
add(7, -5, myFunction)
```

```
12
```

- 常用高阶函数

map/reduce: map将传入的函数依次作用到序列的每个元素，并把结果作为新的list返回；reduce把一个函数作用在一个序列[x1, x2, x3...]上，这个函数必须接收两个参数，reduce把结果继续和序列的下一个元素做累积计算

```
myList = [-1, 2, -3, 4, -5, 6, 7]
map(abs, myList)
```

```
<map at 0x15afaa00630>
```

```
from functools import reduce
def powerAdd(a, b):
    return pow(a, 2) + pow(b, 2)

reduce(powerAdd, myList) # 是否是计算平方和?
```

```
356002059820563014529693 8
```

filter：filter()把传入的函数依次作用于每个元素，然后根据返回值是True还是False决定保留还是丢弃该元素

```
def is_odd(x):
    return x % 3  # 0被判断为False，其它被判断为True

filter(is_odd, myList)
```

```
<filter at 0x15afa9f0588>
```

sorted: 实现对序列排序，默认情况下对于两个元素x和y，如果认为x < y，则返回-1，如果认为x == y，则返回0，如果认为x > y，则返回1

默认排序：数字大小或字母序（针对字符串）

```
sorted(myList)
```

```
[-5, -3, -1, 2, 4, 6, 7]
```

*练习：自定义一个排序规则函数，可将列表中字符串忽略大小写地，按字母序排列，列表为['Apple', 'orange', 'Peach', 'banana']。提示:字母转换为大写的方法为some_str.upper()，转换为小写使用some_str.lower()

- 返回函数: 高阶函数除了可以接受函数作为参数外，还可以把函数作为结果值返回

```
def powAdd(x, y):
    def power(n):
        return pow(x, n) + pow(y, n)
    return power

myF = powAdd(3, 4)
myF
```

```
<function __main__.powAdd.<locals>.power>
```

```
myF(2)
```

```
25
```

- 匿名函数：高阶函数传入函数时，不需要显式地定义函数，直接传入匿名函数更方便

```
f = lambda x: x * x
f(4)
```

```
16
```

等同于：

```
def f(x):
    return x * x
```

```
map(lambda x: x * x, myList)
```

```
<map at 0x15afaa1d0f0>
```

匿名函数可以传入多个参数

```
reduce(lambda x, y: x + y, map(lambda x: x * x, myList))
```

```
140
```

返回函数可以是匿名函数

```
def powAdd1(x, y):
    return lambda n: pow(x, n) + pow(y, n)

lamb = powAdd1(3, 4)
lamb(2)
```

25

其它

- 标识符第一个字符只能是字母或下划线，第一个字符不能出现数字或其他字符；标识符除第一个字符外，其他部分可以是字母或者下划线或者数字，标识符大小写敏感，比如name跟Name是不同的标识符。
- Python规范：
- 类标识符每个字符第一个字母大写；
- 对象\变量标识符的第一个字母小写，其余首字母大写，或使用下划线'_'连接；
- 函数命名同普通对象。
- 关键字

关键字是指系统中自带的具备特定含义的标识符

```
# 查看一下关键字有哪些，避免关键字做自定义标识符
import keyword
print (keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'e
```

- 注释

Python中的注释一般用#进行注释

- 帮助

Python中的注释一般用？查看帮助

往期精彩回顾

- 那些年做的学术公益-你不是一个人在战斗
- 适合初学者入门人工智能的路线及资料下载
- 机器学习在线手册
- 深度学习在线手册

备注：加入本站微信群或者qq群，请回复“加群”

加入知识星球（4500+用户，ID：92416895），请回复“知识星球”