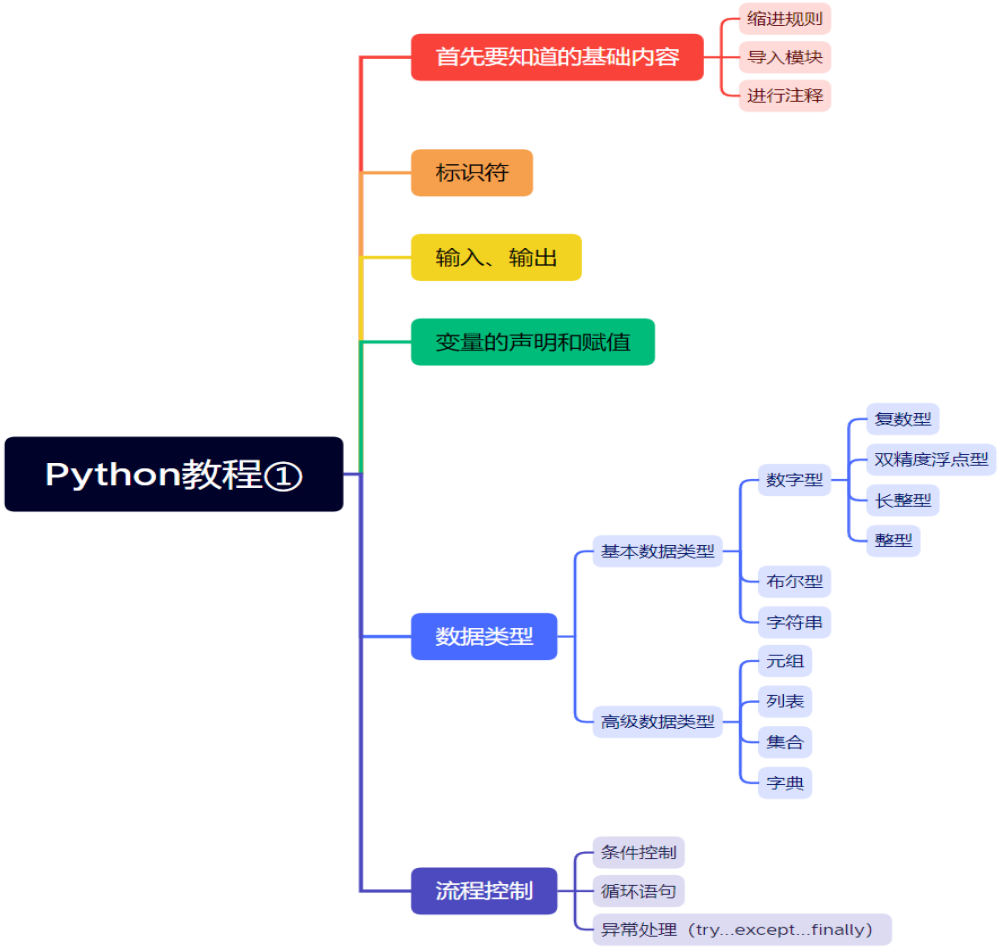
目录


Python教程①
首先要知道的基础内容
缩进规则
导入模块
进行注释
标识符
输入、输出
变量的声明和赋值
数据类型
基本数据类型
数字型
复数型
双精度浮点型
长整型
整型
布尔型
字符串
高级数据类型
元组
列表
集合
字典
流程控制
条件控制
循环语句
异常处理（try...except...finally）

首先要知道的

- Python中的缩进规则
- 如何导入模块
- 如何进行注释

Python中的缩进规则

```
if   False:
      print ("one")
      print ("two")
else:
print( "three") # 缩进没有对齐，在执行时会报错
   print ("four")
```

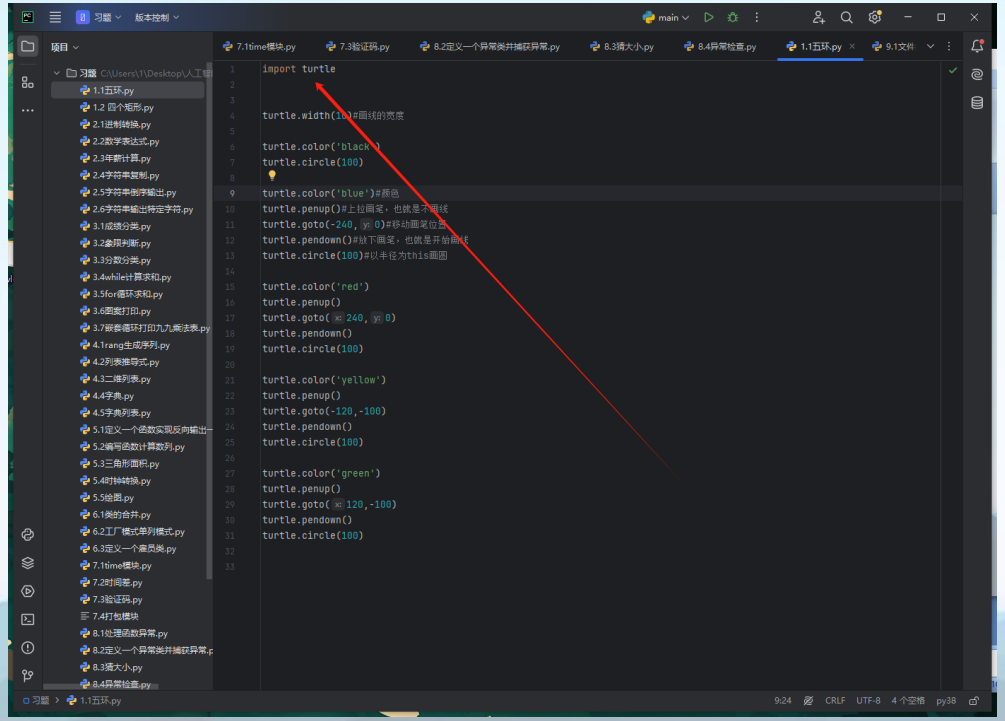
如上所示，python以缩进来判断代码的归属

一般用2-4个空格，tab是四个空格。

同一代码块中缩进空格数需要一致，不能混用

Python的缩进规则是十分严格的
如果代码报错十有八九是缩进的问题
所以先检查缩进是否正确

模块的导入



使用import语句导入模块

类比C中的include

如何注释

- 使用#进行单行注释，类比C中的//

- 使用”””注释内容”””或'''注释内容'''进行多行注释

  （在python中并没有双引号和单引号之分）

标识符

介绍：

标识符主要用于对变 量、函数、类、模块等的命名。

标识符的规则

标识符有如下特定的规则：

- 区分大小写。如：sxt和SXT是不同的
- 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
- 不能使用关键字。比如：if、or、while等。

- _单下划线开头：用于表示某个变量、方法或属性是内部使用的或具有特定含义，但不希望在外部直接访问

- __双下划线开头：类的私有成员。

- __xxx__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。比如：__init__ 是类的构造函数。

Python的保留字

不能随便用

Python 保留字不能作为常数或变量，以及其它任何标识符名称，且都是小写字母。

表 2-2　Python 保留字

and	def	exec	if	not	return
assert	del	finally	import	or	try
break	elif	for	in	pass	while
class	else	from	is	print	with
continue	except	global	lambda	raise	yield

1.1print输出

【示例 2-6】使用 print 打印”Hello Python !”

```
print("Hello Python")
```

执行结果如图 2-5 所示：

【示例 2-7】使用 print 打印一首<<静夜思>>

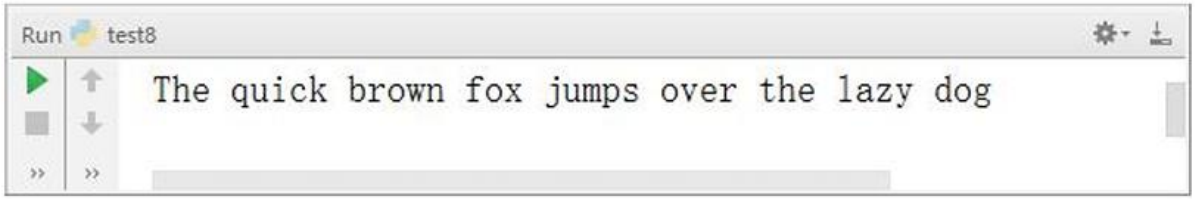
```
#输出多行字符串
print('''
        静夜思
         李白
    床前明月光,
    疑是地上霜。
    举头望明月,
    低头思故乡。
''')
```

执行结果如图 2-6 所示：

【示例 2-8】使用 print 打印多个字符串

```
#print 接收多个参数
print("The quick brown fox","jumps over","the lazy dog")
```

执行结果如图 2-7 所示：



左侧是用三个双引号的多行输出，右侧是字符串合并输出

【示例 2-9】使用 print()对表达式进行美化

```
#print()对表达式进行美化
print("300+200",300+200)
```

执行结果如图 2-8 所示：

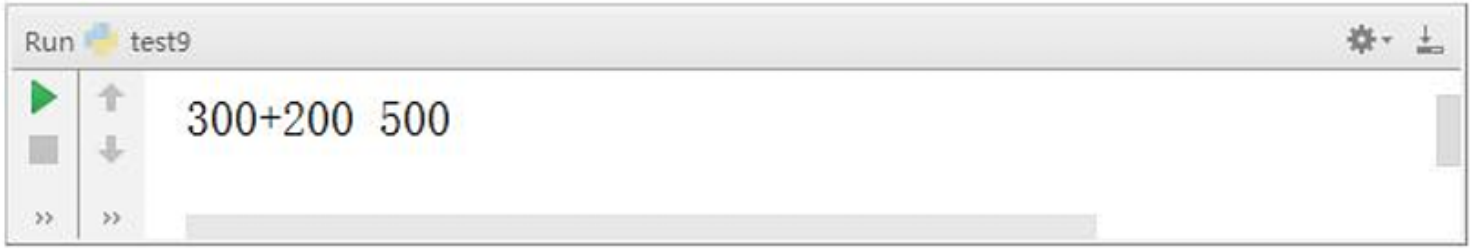


图 2-8 示例 2-9 执行结果

注意：

- 对于 100 + 200，Python 解释器自动计算出结果 300，但是'100 + 200 ='是字符串而非数学公式，Python 把它视为字符串

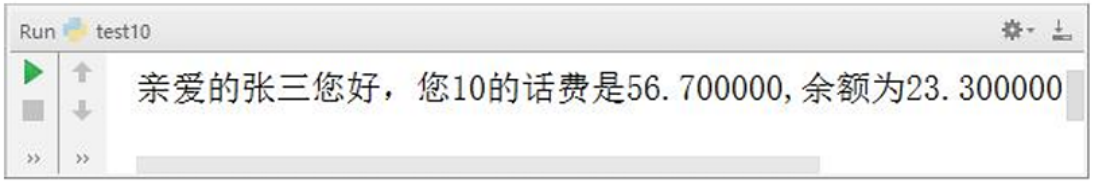
表 2-3 Python 占位符

占位符	类型	示例
%d	整数	a=10,b=5 , print(“%d-%d=%d” %(a,b,(a-b)))
%f	浮点数	a=3.5 print(“大白菜%0.1f 一斤” %a)
%s	字符串	name=” admin” ;print(“我叫%s” %name)
%x	十六进制整数	a=20 print(“%x” %a)

【示例 2-10】使用%实现格式化输出

```
#格式输出
name="张三"
month=10
change=56.7
balance=23.3
print("亲爱的%s 您好，您%d 的话费是%f,余额为%f"%(name,month,change,balance))
```

执行结果如图 2-9 所示：



占位符

和C差不多

不过由 ,#name,#month

变为了 %(name,month)

1.2input输入

```
#input 输入
userName=input('请输入你的姓名:')
age=int(input('请输入你的年龄:'))
sal=float(input('请输入你的薪资:'))
print('姓名是%s,年龄是%d,薪资是%f'%(userName,age,sal))
```

需要注意的是，input获取的是字符串
所以赋值给变量前记得强制转换

还有不同于C
3.x版本python强制转换是个函数

执行结果如图 2-10 所示：

图 2-10 示例 2-11 执行结果

注意：

- 由于年龄和薪资都是数值,所以在获取用户输入值后,需要分别使用 int 和 float 函数将 Input 函数的返回值转换为整数和浮点数。

1.3变量声明和赋值

```
变量名 = 变量值
```

最简单的表达式就是字面量。比如：a = 123 。 运行过程中，解释器先运行右边的表达式，生成一个代表表达式运算结果的对象，然后，将这个对象地址赋值给左边的变量。

在 Python 语言中，声明变量的同时需要为其赋值，毕竟不代表任何值的变量毫无意义，Python 中也不允许有这样的变量。

链式赋值

```
变量名 1=变量名 2=  变量名 3...=变量值
```

系列解包赋值

```
变量名 1,变量名 2,  变量名 3...=变量值 1,变量值 2,变量 3....
```

```
#实现变量交换
b,a=a,b
```

删除变量

```
del  变量名
```

1.4数据类型

基本数据类型	Number	Integer（整型）	
		Long integer（长整型）	
		Double-precision floating（双精度浮点型）	
		Complex number（复数型）	
	Boolean	True 或 False（布尔型）	
	String	零个或多个字符组成的有限序列（字符串型）	Sequence 类型簇
高级数据类型	Tuple	内部元素不可修改（元组型）	Sequence 类型簇
	List	但内部元素可以修改（列表型）	Sequence 类型簇
	Set	内部元素相互之间无序的一组对象（集合型）	
	Dictionary	拥有键/值对的特殊列表，形式 key:value（字典型）	

基本数据类型

Number	Integer（整型）
	Long integer（长整型）
	Double-precision floating（双精度浮点型）
	Complex number（复数型）

```
In [3]: z = 2 +3j
        print(z)
        print(type(z))
```

```
(2+3j)
<class 'complex'>
```

除了复数其他和C大差不差

Boolean　　True 或False（布尔型）,有时候能当作1或0

String

(2) 字符串的序号

代码实例

```
In [1]: str = 'abcdef12345'
        str[0]
Out[1]: 'a'

In [2]: str[1]
Out[2]: 'b'

In [3]: str[-1]
Out[3]: '5'
```


类比C中的字符数组
不过C中没有反向索引

单双引号都行，python无限制

高级数据类型

Tuple	内部元素不可修改（元组型）
List	但内部元素可以修改（列表型）
Set	内部元素相互之间无序的一组对象（集合型）
Dictionary	拥有键/值对的特殊列表，形式 key:value（字典型）

List列表

列表的数据项不需要具有相同的类型

创建一个列表，只要把逗号分隔的不同的数据项使用方括号括起来即可。

```
list1 = ['Google', 'Runoob', 1997, 2000]
list2 = [1, 2, 3, 4, 5 ]
list3 = ["a", "b", "c", "d"]
list4 = ['red', 'green', 'blue', 'yellow', 'white', 'black']
```

类比C中的字符串数组

列表都可以进行的操作包括索引，切片，加，乘，检查成员。
此外，Python 已经内置确定序列的长度以及确定最大和最小的元素的方法。
只需要调用就行。

List常用的脚本操作符

Python 表达式	结果	描述
len([1, 2, 3])	3	长度
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	组合
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	重复
3 in [1, 2, 3]	True	元素是否存在于列表中
for x in [1, 2, 3]: print(x, end=" ")	1 2 3	迭代

List常用函数&方法

函数

len(list)

列表元素个数

max(list)

返回列表元素最大值

min(list)

返回列表元素最小值

list(seq)

将元组转换为列表

方法

list.append(obj)

在列表末尾添加新的对象

list.count(obj)

统计某个元素在列表中出现的次数

list.extend(seq)

在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表）

list.index(obj)

从列表中找出某个值第一个匹配项的索引位置

list.insert(index, obj)

将对象插入列表

list.pop([index=-1])

移除列表中的一个元素（默认最后一个元素），并且返回该元素的值

list.remove(obj)

移除列表中某个值的第一个匹配项

list.reverse()

反向列表中元素

list.sort(key=None, reverse=False)

对原列表进行排序

list.clear()

清空列表

list.copy()

复制列表

Tuple元组

- Python 的元组与列表类似，不同之处在于元组的元素不能修改。
- 元组创建很简单，只需要在括号中添加元素，并使用逗号隔开即可。

切片表达法

```
sequence[start:stop:step]
```

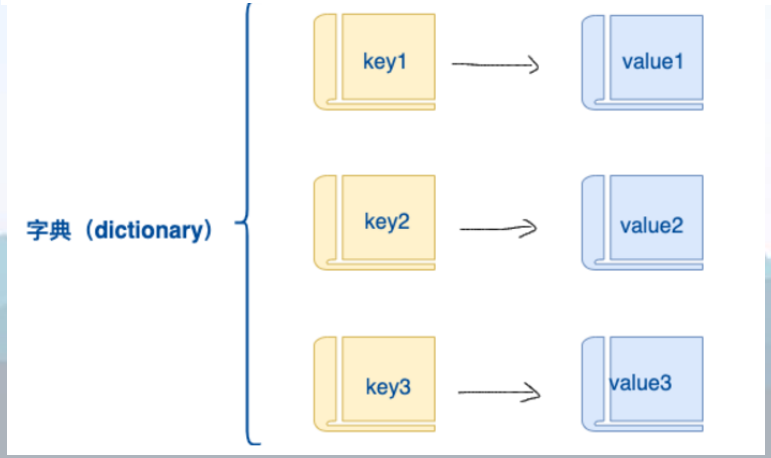
start：切片的起始索引（包含该索引对应的元素）。如果省略，默认为0。
stop：切片的结束索引（不包含该索引对应的元素）。如果省略，默认为序列的长度。
step：步长，表示取元素的间隔。如果省略，默认为1。

```
my_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
sub_list = my_list[2:5]  # 从索引2开始到索引5之前
# sub_list = [2, 3, 4]
```

Dictionary字典

- 字典是另一种可变容器模型，且可存储任意类型对象。
- 字典的每个键值对 (**key=>value**)用冒号 **:** 分割，每个对之间用逗号(**,**)分割，整个字典包括在花括号 **{}** 中

```
d = {key1 : value1, key2 : value2, key3 : value3 }
```



类比C中的指针

键必须是唯一的，但值则不必。
值可以取任何数据类型，但键必须是不可变的，如字符串，数字。

访问字典里的值

把相应的键放入到方括号中

```
#!/usr/bin/python3

tinydict = {'Name': 'Runoob', 'Age': 7, 'Class': 'First'}

print ("tinydict['Name']: ", tinydict['Name'])
print ("tinydict['Age']: ", tinydict['Age'])
```

修改值，添加新的元素

```
#!/usr/bin/python3

tinydict = {'Name': 'Runoob', 'Age': 7, 'Class': 'First'}

tinydict['Age'] = 8  # 更新 Age
tinydict['School'] = "QUT"  # 添加信息

print("tinydict['Age']: ", tinydict['Age'])
print("tinydict['School']: ", tinydict['School'])
```

删除元素或字典

```
del tinydict['Name'] # 删除键 'Name'
tinydict.clear()     # 清空字典
```

清空了之后不能再删除，
因为字典已经没了

字典内置函数&方法

函数及描述	实例

len(dict)
计算字典元素个数，即键的总数。
	`>>> tinydict = {'Name': 'Runoob', 'Age': 7, 'Class': 'First'}`
	`>>> len(tinydict)`
	`3`

str(dict)
输出字典，可以打印的字符串表示。
	`>>> tinydict = {'Name': 'Runoob', 'Age': 7, 'Class': 'First'}`
	`>>> str(tinydict)`
	`"{'Name': 'Runoob', 'Class': 'First', 'Age': 7}"`

type(variable)
返回输入的变量类型，如果变量是字典就返回字典类型。
	`>>> tinydict = {'Name': 'Runoob', 'Age': 7, 'Class': 'First'}`
	`>>> type(tinydict)`
	`<class 'dict'>`

函数及描述
dict.clear() 删除字典内所有元素
dict.copy() 返回一个字典的浅复制
dict.fromkeys() 创建一个新字典，以序列seq中元素做字典的键，val为字典所有键对应的初始值
dict.get(key, default=None) 返回指定键的值，如果键不在字典中返回 default 设置的默认值
key in dict 如果键在字典dict里返回true，否则返回false
dict.items() 以列表返回一个视图对象
dict.keys() 返回一个视图对象
dict.setdefault(key, default=None) 和get()类似, 但如果键不存在于字典中，将会添加键并将值设为default
dict.update(dict2) 把字典dict2的键/值对更新到dict里
dict.values() 返回一个视图对象
pop(key[,default]) 删除字典 key（键）所对应的值，返回被删除的值。
popitem() 返回并删除字典中的最后一对键和值。

Set集合

集合（set）是一个无序的不重复元素序列。
集合中的元素不会重复，并且可以进行交集、并集、差集等常见的集合操作。
可以使用大括号 **{ }** 创建集合，元素之间用逗号 **,** 分隔， 或者也可以使用 **set()** 函数创建集合

```
>>> a = set('abracadabra')

>>> a
{'a', 'r', 'b', 'c', 'd'}
```

常用集合内置方法

添加元素	s.add(x)　　s.update(x)
移除元素	s.remove(x)　　s.discard(x)
计算集合元素个数	len(s)
清空集合	s.clear()
判断元素是否在集合中存在	x in s

方法	描述
add()	为集合添加元素
clear()	移除集合中的所有元素
copy()	拷贝一个集合
difference()	返回多个集合的差集
difference_update()	移除集合中的元素，该元素在指定的集合也存在。
discard()	删除集合中指定的元素
intersection()	返回集合的交集
intersection_update()	返回集合的交集。

isdisjoint()	判断两个集合是否包含相同的元素，如果没有返回 True，否则返回 False。
issubset()	判断指定集合是否为该方法参数集合的子集。
issuperset()	判断该方法的参数集合是否为指定集合的子集
pop()	随机移除元素
remove()	移除指定元素
symmetric_difference()	返回两个集合中不重复的元素集合。
symmetric_difference_update()	移除当前集合中在另外一个指定集合相同的元素，并将另外一个指定集合中不同的元素插入到当前集合中。
union()	返回两个集合的并集
update()	给集合添加元素
len()	计算集合元素个数

1.5流程控制

- 条件控制
- 循环语句
- 异常处理（**try...except...finally**）

条件控制

if 语句

```
if condition_1:
    statement_block_1
elif condition_2:
    statement_block_2
else:
    statement_block_3
```

如果cond1条件为True，执行block_1,否则如果条件cond2条件为True，执行block_2,否则执行block_3

match...case

```
match subject:
    case <pattern_1>:
        <action_1>
    case <pattern_2>:
        <action_2>
    case <pattern_3>:
        <action_3>
    case _:
        <action_wildcard>
```

Py3.10以上才有的东西

用法类比C中的switch..case

循环语句

while 循环

```
while 判断条件(condition)：
    执行语句(statements)……
```

for 语句

```
for <variable> in <sequence>:
    <statements>
else:
    <statements>
```

```
sites = ["Baidu", "Google","Runoob","Taobao"]
for site in sites:
    print(site)
```

以上代码执行输出结果为：

```
Baidu
Google
Runoob
Taobao
```

Python的for比C中的更灵活些多了一种“索引”的感觉

range() 函数

如果你需要遍历数字序列，可以使用内置 range() 函数。它会生成数列

```
>>>for i in range(5):
...     print(i)
...
0
1
2
3
4
```

```
range(start, stop[, step])
```

- start: 计数从 start 开始。默认是从 0 开始。例如range（5）等价于range（0， 5）；
- stop: 计数到 stop 结束，**但不包括 stop**。例如：range（0， 5） 是[0, 1, 2, 3, 4]没有5
- step：步长，默认为1。例如：range（0， 5） 等价于 range(0, 5, 1)

推导式

```
[expression for item in iterable if condition]
```

expression：对 item 进行操作后得到的值，会成为新列表的元素。
item：从 iterable 中迭代得到的元素。
iterable：一个可迭代对象，比如列表、元组、字典等。
condition：一个可选的条件表达式，只有当条件为真时，expression 才会被添加到结果列表中。

```
squares = [x**2 for x in range(10)]
# squares = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

很像C中for的三个参数