Python中的“瓶瓶罐罐”

实验指导手册

版本:1.0

华为技术有限公司

版权所有 © 华为技术有限公司 2020。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明


和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://edu.huaweicloud.com/

目录

华为专有和保密信息

版权所有 © 华为技术有限公司

背景介绍

Python 是一个高层次的结合了解释性、编译性、互动性和面向对象的脚本语言，由于其简单的语法、强大的工具库等优势被很多人青睐，应用于各个行业之中。

实验目的

本实验指导主要实了解 Python 中不同数据类型的使用和运算符的使用。通过本实验，您将能够：

- 了解 Python 中多种结构的使用
- 了解 Python 运算符的使用

1 Python 中数据类型的使用

1.1 实验介绍

实验介绍了本对于Python中六种常用数据类型的使用、Python中运算符的使用以及Python中数据拷贝的实现。

1.2 实验目的

通过以下小的实验可以帮助我们掌握Python这门编程语言各种数据类型的使用、运算符的使用和数据拷贝的实现。

1.3 资源准备

安装 Python 解释器/anaconda。

安装好 jupyter lab。

（方便起见，实验所使用的环境为云上环境 modelarts）

1.4 数值

步骤 1　新建 notebook（python3）文件。并修改名称为 Python 数据类型的使用。

步骤 2　创建不同类型的数值数据

```
num_int = 1
num_float = 1.0
number_com= 1+1j
```

步骤 3　数值的运算

```
print(5/2)            # 输出 2.5
print(5//2)           # 输出 2，//为取整运算符
print(5%2)            # 输出 1，%为取余运算符
```

```
print(3**2)             # 输出 9，**表示乘方操作
print(5+1.6)            # 输出 6.6，不同精度的类型的数字相加默认取高精度类型作为结果
```

步骤 4　内置方法的使用

```
abs(-1.1)   # 输出 1.1
round(1.1)  # 输出 1
round(-1.6) # 输出-2
divmod(5,2) # 输出(2, 1)
max(1,2,3,5)# 输出 5
```

步骤 5　math 模块的使用

```
math.cos(math.pi) # 输出-1.0，求取 math.pi 的余弦值
math.fabs(-1)     # 输出 1.0,会将结果转化为浮点数
math.ceil(1.1)    # 输出 2，向上取整
math.floor(1.1)   # 输出 1，向下取整
math.factorial(4) # 输出 24，求阶乘
hello world
```

步骤 6　bool 类型的运算

```
print(True+True)
print(True+False)
print(False-True)
print(True*5)
```

输出：

```
2
1
-1
5
```

1.5 字符串

步骤 1　使用不同的方式创建字符串

单引号和双引号创建字符串：

```
str1 = "python"
str2 = 'python'
print(str1)
print(str2)
```

输出：

```
python
```

```
python
```

三引号创建字符串：

```
str3 = """
我这个字符串太长了，
还要换个行。
"""
str4 = '''
我这个字符串太长了，
还要换个行。
'''
print(str3)
print(str4)
```

输出：

```
我这个字符串太长了，
还要换个行。

我这个字符串太长了，
还要换个行。
```

步骤 2　使用 len 函数查看字符串长度

```
len(str1)
```

输出：

```
6
```

步骤 3　这是一些 Python 解释器不认识的字符串

```
s = "
python
"
```

```
  File "<ipython-input-24-9f7ad361d69a>", line 1
    s = "
        ^
SyntaxError: EOL while scanning string literal
```

同理，以下也会报错：

```
s = "张三说："这是一个字符串""
s = "张三说：这是一个字符串'
```

步骤 4　转义字符和原始字符串的使用

```
s = "张三说：\"这是一个字符串\""
print(s)
s = "张三说：\n'\\这是一个字符串'"
print(s)
```

```
s = r"张三说：\n'\\这是一个字符串'"
print(s)
```

输出：

```
张三说："这是一个字符串"
张三说：
'\这是一个字符串'
张三说：\n'\\这是一个字符串'
```

步骤 5　字符串的运算

```
str1 = "python"
str2 = "PYTHON"
print(str1+str2)
print(str1*3)
```

输出：

```
pythonPYTHON
pythonpythonpython
```

步骤 6　索引

```
# 使用索引获取字符串中的元素
str1 = "python"
print(str1[0],str1[1],str1[2],str1[3],str1[4],str1[5])
print(str1[-1],str1[-2],str1[-3],str1[-4],str1[-5],str1[-6])
```

输出：

```
p y t h o n
n o h t y p
```

步骤 7　切片

```
# 使用切片获取字符串中的元素
str1 = "python"
# 取所有
print("str1[::]：",str1[::])
print("str1[::2]：",str1[::2])
print("str1[2::2]：",str1[2::2])
print("str1[1:5:2]：",str1[1:5:2])
print("str1[-1:-5:2]：",str1[-1:-5:2])
print("str1[-1:-5:-2]：",str1[-1:-5:-2])
```

输出：

```
str1[::]： python
str1[::2]： pto
str1[2::2]： to
```

```
str1[1:5:2]:  yh
str1[-1:-5:2]:
str1[-1:-5:-2]:  nh
```

步骤 8　字符串的不可变性

字符串不可变，强制修改会报错：

```
str1 = "python"
# 修改 p 为 c
str1[0] = "c"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-1-b144729dce58> in <module>
      1 str1 = "python"
      2 # 修改p为c
----> 3 str1[0] = "z"

TypeError: 'str' object does not support item assignment
```

步骤 9　字符串的内置方法

```
S = "python"                                    # 变量赋值
# str.split(str="", num=-1)：通过指定分隔符对字符串进行切片，如果参数 num 有指定值，则分隔 num+1 个子字符串，-1 表示分割所有。
print(S.split('h'))                             # 输出['pyt','on']，根据 h 对字符串切割
# str.replace(old, new[, max])：返回字符串中的 old（旧字符串） 替换成 new(新字符串)后生成的新字符串，如果指定第三个参数 max，则替换不超过 max 次。
print(S.replace('py','PY'))                     # Python，将字符串中的 py 替换为 PY
# str.upper()：返回小写字符转化为大写后的值。
print(S.upper())                                # PYTHON
# str.lower()：返回大写字符转化为小写后的值。
print('PYTHON'.lower())                         # python，字符串转小写
line='aa,bb,ccc,dd\n'                           # \n 为换行
# str.join(sequence)：sequence：要连接的序列，返回指定字符连接序列中元素后生成的新字符串。
print(''.join(['life', 'is' ,'short']))         # 输出 life is short，join 拼接字符串
```

输出：

```
['pyt', 'on']
PYthon
PYTHON
python
lifeisshort
```

检测某个字符串是否包含在另一个字符串中中，如果是返回开始的索引值，否则返回-1：

```
mystr = 'hello world, my name is python'
str1 = "python"
# str.find (str1, beg=0, end=len(string))，查看是否包含 str，可以通过 beg 和 end 指定查找范围，返回索引位置，找不到返回-1
```

```
mystr.find(str1,0,len(mystr)) # 输出 23
mystr.find(str1,5,20)          # 输出-1
```

统计给定字符的出现次数：

```
mystr = 'hello world, my name is python'
# str.count(str1, beg=0, end=len(string)),在指定范围统计给定字符出现的次数
mystr.count("n", 0, len(mystr)) # 输出 2
```

首字母变为大写：

```
mystr = 'hello world, my name is python'
print(mystr.capitalize()) # 第一个首字母大写
print(mystr.title()) # 所有单词首字母大写
```

输出：

```
Hello world, my name is python
Hello World, My Name Is Python
```

查看字符串是否以某个字符开始或结束：

```
mystr = 'hello world, my name is python'
# str.startswith(s)：是否以 s 开始
print(mystr.startswith("hello"))
# str. endswith(s)：是否以 s 结束
print(mystr.endswith("world"))
```

输出：

```
True
False
```

删除字符串中开始和结尾的空格：

```
mystr = '   hello world, my name is python   '
# str.strip()：删除两边的空格
print(mystr.strip())
```

检查字符串中的数字和字母：

```
mystr = 'hello world, my name is python996'
# str.isalpha:查看是否全是字母
print(mystr.isalpha())
# str.isalpha:查看是否全是数字
print(mystr.isdigit())
# str.isalnum：查看是否是字母和数字组成的
print(mystr.isalnum())
```

输出：

```
False
False
```

```
False
```

步骤 10　格式化输出

格式化输出，是指将指定的数据按照给定的格式进行输出。

```
# 格式化操作符（百分号%）、字符串转换类型和格式化操作符辅助指令实现格式化输出
'My name is %s , age is %d' %('AI', 63)
```

输出：

```
'My name is AI , age is 63'
```

在输出时，设置输出值的格式：

```
print('%f' % 1.11)  # 默认保留 6 位小数
print('%.1f' % 1.11)  # 取 1 位小数
print('%e' % 1.11)  # 默认 6 位小数，用科学计数法
print('%.3e' % 1.11)  # 取 3 位小数，用科学计数法
print('%20s' % 'hello world')  # 右对齐，取 20 位，不够则补位
print('%-20s' % 'hello world')  # 左对齐，取 20 位，不够则补位
print('%.2s' % 'hello world')  # 取 2 位
```

输出：

```
1.110000
1.1
1.110000e+00
1.110e+00
        hello world
hello world
he
```

其他形式的格式化输出：

```
# str.format
mystr = '{} world, my name is {}'
print(mystr.format("hello","python"))
# 指定位置
mystr = '{0} world, my name is {1}'
print(mystr.format("hello","python"))
# f, Python3.6 以后的版本支持
str1 = "python"
mystr = f'hello world, my name is {str1}'
print(mystr)
```

输出：

```
hello world, my name is python
hello world, my name is python
hello world, my name is python
```

1.6 元组

步骤 1 创建一个元组

```
t1 = (1,2,3)
t2 = 1,2,3
print(type(t1))
print(type(t2))
```

输出:

```
<class 'tuple'>
<class 'tuple'>
```

步骤 2 修改元组中的内容

```
t1 = (1,2,3)
t1[0] = 5
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-28-305bfd606f67> in <module>()
      1 t1 = (1,2,3)
----> 2 t1[0] = 5

TypeError: 'tuple' object does not support item assignment
```

步骤 3 单元素元组

```
t1 = (1)
t2 = (1,)
t3 = ((1))
t4 = ((1,))
print(type(t1))
print(type(t2))
print(type(t3))
print(type(t4))
```

输出:

```
<class 'int'>
<class 'tuple'>
<class 'int'>
<class 'tuple'>
```

步骤 4 元组的运算

```
t1 = (1,2,3)
print(t1+t1)
print(t1*3)
```

输出:

```
(1, 2, 3, 1, 2, 3)
(1, 2, 3, 1, 2, 3, 1, 2, 3)
```

1.7 列表

步骤 1　创建一个列表

```
l1 = [1,2,3]
l2 = list("python")
print(l2)  # 输出: ['p', 'y', 't', 'h', 'o', 'n']
```

步骤 2　列表推导式

```
[i for i in range(5)]   # 输出: [0, 1, 2, 3, 4]
[i for i in range(5) if i%2] # 输出: [1, 3]
```

步骤 3　索引和切片

```
l = [1,2,3,4,5]
print(l[0]) # 1
print(l[::2]) # [1, 3, 5]
```

步骤 4　常用操作

内置方法的使用：

```
animals = ['cat', 'dog', 'monkey']
# list.append(obj)：在列表末尾添加新的对象。
animals.append('fish')                  # 追加元素
print(animals)                          # 输出 ['cat', 'dog', 'monkey', 'fish']
# list.remove(obj)：移除列表中某个值的第一个匹配项。
animals.remove('fish')                  # 删除元素 fish
print(animals)                          # 输出 ['cat', 'dog', 'monkey']
# list.insert(index, obj)：用于将指定对象插入列表的指定位置。index：插入位置
animals.insert(1,'fish')                # 在下标 1 的地方插入元素 fish
print(animals)                          # 输出 ['cat', 'fish', 'dog', 'monkey']
# list.pop([index=-1])：要移除列表中对下标对应的元素（默认是最后一个）。Index：下标
animals.pop(1)                          # 删除下标为 1 的元素
print(animals)                          # 输出 ['cat', 'dog', 'monkey']
```

获取数据和对应下标：

```
animals = ['cat', 'dog', 'monkey']
# enumerate(sequence) ：将一个可遍历的数据对象组合为一个索引序列，同时列出数据和数据下标，一般用在 for 循环当中。
for i in enumerate(animals):
    print(i)                            # 元素下标和元素所组成的索引
```

输出：

```
(0, 'cat')
```

```
(1, 'dog')
(2, 'monkey')
```

列表数据排序：

```
list1 = [12,45,32,55]
# list.sort(cmp=None, key=None, reverse=False)：cmp 为可选参数，如果指定了该参数，会使用该参数的方法进行排序。key 是用来进行比较的元素。reverse 为排序规则，False 为升序。
list1.sort()                    # 对列表进行排序
print(list1)                    # 输出[12,32,45,55]
```

逆序列表中的元素：

```
# list.reverse()：反向列表中元素。
list1.reverse()                 # 对列表进行逆置
print(list1)                    # 输出[55,45,32,12]
```

查找与统计：

```
list1 = [1,23,3,4,1]
# 和字符串一样使用 index 和 count 方法
print(list1.index(1)) # 0
print(list1.count(1)) # 2
```

步骤 5　可以被修改的元组

```
t = (1,2,[3,4])
t[2][0] = 6
print(t)
```

输出：

```
(1, 2, [6, 4])
```

当元组中的元素为列表这类可变元素时，可以进行修改，但是修改只限于可变元素的修改，而非元组。

1.8 字典

步骤 1　创建一个字典

```
# 字典的三种赋值操作
# 字典的三种赋值操作
x = {'a':'A','b':"B",'c':3}
X = dict(a='A',b="B", c=3)
x = dict([("a", "A"),("b", "B"),("c",3)])
```

步骤 2　字典推导式

```
dict1={"a":1, "b":2, "c":3, "d":4}
```

```
dict2={v:k for k,v in dict1.items()}
print(dict2)
```

输出：

```
{1: 'a', 2: 'b', 3: 'c', 4: 'd'}
```

步骤 3 字典的错误操作

```
x = {'a':'A','b':"B",'c':3,"a":1} # {'a': 1, 'b': 'B', 'c': 3}，值被覆盖
x = {'a':'A','b':"B",'c':3,[1,2]:"AB"} # 键不可变，报错
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-988c55dea538> in <module>
----> 1 x = {'a':'A','b':"B",'c':3,[1,2]:"AB"}

TypeError: unhashable type: 'list'
```

```
x[1] # 字典无序不能使用下标
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-14-fb0e0080324e> in <module>
----> 1 x[1]

KeyError: 1
```

步骤 4 获取字典中的数据

```
x = {'a': 1, 'b': 'B', 'c': 3}
# 使用键获取值
print(x["a"])
print(x["x"]) # 如果访问不存在的键，会报错
```

```
1
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-16-cdf999953ce6> in <module>
      2 # 使用键获取值
      3 print(x["a"])
----> 4 print(x["x"]) # 如果访问不存在的键，会报错

KeyError: 'x'
```

```
# 使用 get(key,["str"])方法获取数据
print(x.get("a"))
print(x.get("x")) # 使用 get 方法获取数据时，如果键不存在会返回指定的值 str
print(x.get("x","x 不存在"))
```

输出：

```
1
None
x 不存在
```

获取所有的键值对信息：

```
# 查看所有的键
print(x.keys())
# 查看所有的值
print(x.values())
```

```
# 查看所有的键值对
print(x.items())
```

输出：

```
dict_keys(['a', 'b', 'c'])
dict_values([1, 'B', 3])
dict_items([('a', 1), ('b', 'B'), ('c', 3)])
```

步骤 5　修改、添加和删除

```
# 向字典中插入数据
x["x"] = "x"
print(x)
# 修改字典中的数据
x["a"] = "A"
print(x)
# 删除字典中的数据
x.pop("x")
print(x)
```

输出：

```
{'a': 'A', 'b': 'B', 'c': 3, 'x': 'x'}
{'a': 'A', 'b': 'B', 'c': 3, 'x': 'x'}
{'a': 'A', 'b': 'B', 'c': 3}
```

清空字典：

```
x.clear()
print(x) # {}
```

1.9 集合

步骤 1　创建一个集合

```
sample_set = {'Prince', 'Techs'}
sample_set = set(['Prince', 'Techs'])
```

步骤 2　常用操作

```
# set.add(obj)：给集合添加元素，如果添加的元素在集合中已存在，则不执行任何操作。
sample_set.add('Data')            # 向集合中增加元素 Data
print(sample_set)                 # 输出 {'Prince', 'Techs', 'Data'}
print(len(sample_set))            # 输出 3
# set.remove(obj)：移除集合中的指定元素。
sample_set.remove('Data')         # 删除元素 Data
print(sample_set)                 # {'Prince', 'Techs'}
```

```
list2 = [1,3,1,5,3]
print(list(set(list2)))          # 输出 [1,3,5]，利用集合元素的唯一性进行列表去重
sample_set = frozenset(sample_set)# 不可变集合
```

1.10 数据拷贝

步骤 1　Python 中引用所在的问题

```
l = [1,2,3,[5,4]]
s = {"a":1,"b":[2,"B"]}
l1 = l
l1[0] = 0
s1 = s
s1["a"] = "A"
print("l:",l)
print("l1:",l1)
print("s:",s)
print("s1:",s1)
```

输出：

```
l: [0, 2, 3, [5, 4]]
l1: [0, 2, 3, [5, 4]]
s: {'a': 'A', 'b': [2, 'B']}
s1: {'a': 'A', 'b': [2, 'B']}
```

步骤 2　浅拷贝

```
l = [1,2,3,[5,4]]
s = {"a":1,"b":[2,"B"]}
l1 = l.copy() # copy 方法实现浅拷贝
l1[0] = 0
s1 = s.copy()
s1["a"] = "A"
print("l:",l)
print("l1:",l1)
print("s:",s)
print("s1:",s1)
```

输出：

```
l: [1, 2, 3, [5, 4]]
l1: [0, 2, 3, [5, 4]]
s: {'a': 1, 'b': [2, 'B']}
s1: {'a': 'A', 'b': [2, 'B']}
```

浅拷贝中存在的问题：

```
l = [1,2,3,[5,4]]
s = {"a":1,"b":[2,"B"]}
```

```
l1 = l.copy()
l1[3][0] = 0
s1 = s.copy()
s1["b"][0] = "b"
print("l:",l)
print("l1:",l1)
print("s:",s)
print("s1:",s1)
```

输出:

```
l: [1, 2, 3, [0, 4]]
l1: [1, 2, 3, [0, 4]]
s: {'a': 1, 'b': ['b', 'B']}
s1: {'a': 1, 'b': ['b', 'B']}
```

步骤 3 深拷贝

```
import copy
l = [1,2,3,[5,4]]
s = {"a":1,"b":[2,"B"]}
l1 = copy.deepcopy(l)
l1[3][0] = 0
s1 = copy.deepcopy(s)
s1["b"][0] = "b"
print("l:",l)
print("l1:",l1)
print("s:",s)
print("s1:",s1)
```

输出:

```
l: [1, 2, 3, [5, 4]]
l1: [1, 2, 3, [0, 4]]
s: {'a': 1, 'b': [2, 'B']}
s1: {'a': 1, 'b': ['b', 'B']}
```

1.11 运算符

步骤 1 比较运算符

```
a = 1
b = 1.0
print(a == b)  # True
print(a != b)  # False
print(a > b) # Fasle
print(a < b) # False
print(a <= b) #True
```

步骤 2 赋值运算符

```
# 赋值运算符
a=5
a+=1
print("a+=1:",a)
a-=1
print("a-=1:",a)
a*=2
print("a*=2:",a)
a/=2
print("a/=2:",a)
```

输出：

```
a+=1: 6
a-=1: 5
a*=2: 10
a/=2: 5.0
```

步骤 3 逻辑运算符

```
print(True and False)
print(True or False)
print(not False)
```

输出：

```
False
True
True
```

步骤 4 成员运算符

```
L = [1,2,4,"a"]
print(1 in L) # True
print("b" not in L) # True
```

1.12 小结任务

当前有以下需求：

实现一个用于计算（包括加减乘除）的小程序：

- 接受用户输入的计算式（如：3*4+1.1）；
- 计算值；
- 将计算过程中涉及到的符号存储在元组中；
- 将计算中涉及到的值去重后存在列表里面（由大到小排列）；

- 最后将计算过程存在字典里面（如 {"3*4+1.1":13.1}）

  提示：

  - input 接受为字符串类型；
  - eval 函数可以用直接计算字符串中的计算式；
  - str.split 可进行字符串切割。