2-1Python基础知识

2020-知数堂-Python第二期

章老师

基础篇

1. 一行代码实现1--100之和
2. 如何在一个函数内部修改全局变量
3. 列出python中可变数据类型和不可变数据类型
4. 简述下*args，**kwargs如何使用
5. 请写出两个函数，实现数字到字符串，字符串到数字的转换，比如10 →”10”, “10” →10
6. Python函数参数是传值还是传址
7. 请问了解str和bytes之间的区别的么

基础题

1. 请简述isinstance和type的区别
2. 会使用lambda表达式么
3. 请问if表达式为假有哪几种情况
4. 请问了解字符串的intern机制么
5. 请问`A or B and C` 是如何计算的
6. 请问了解生成器、迭代器么

基础篇

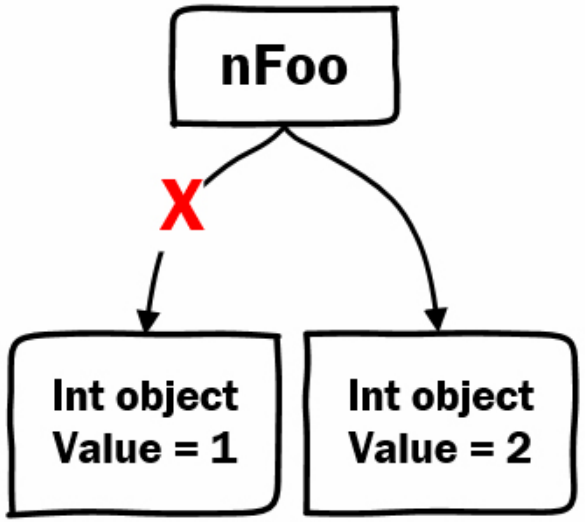
- 一行代码实现1--100之和
  - sum(range(1,n+1)) 考察对内置函数的了解

- 如何在一个函数内部修改全局变量
  - global关键词，全局变量与局部变量

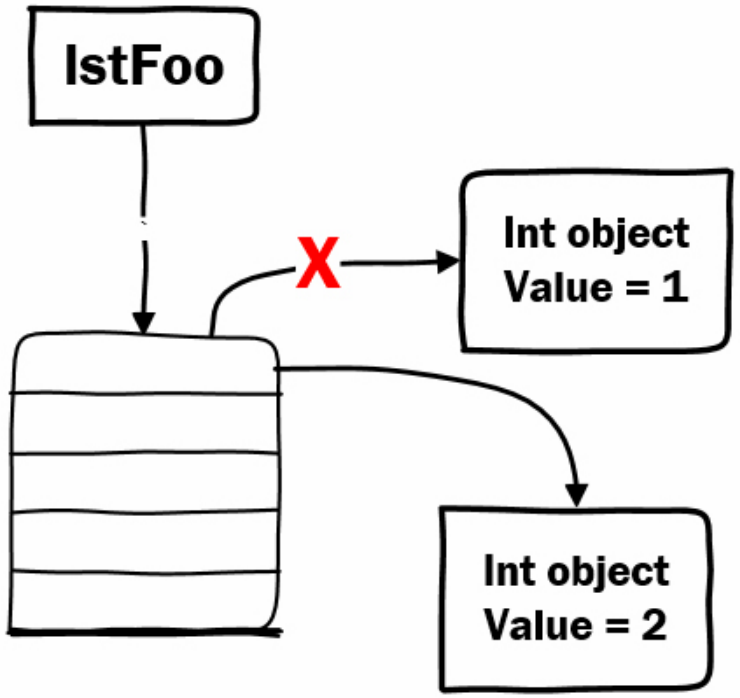
基础篇

- 列出python中可变数据类型和不可变数据类型
  - 不可变：数值型、字符串型string和元组tuple
    - 不允许变量的值发生变化，如果改变了变量的值，相当于是新建了一个对象，而对于相同的值的对象，在内存中则只有一个对象（一个地址），用id()方法可以打印对象的id
  - 可变：列表list和字典dict
    - 允许变量的值发生变化，即如果对变量进行append、+=等这种操作后，只是改变了变量的值，而不会新建一个对象，变量引用的对象的地址也不会变化，

基础篇



修改不可变对象



修改可变对象

基础篇

- 简述下*args，**kwargs如何使用
  - *args表示任何多个无名参数，它本质是一个tuple;
  - **kwargs表示关键字参数，它本质上是一个dict;
  - 怎么用？自己试一下就知道了

- 请写出两个函数，实现数字到字符串，字符串到数字的转换，比如10 →”10”, “10” →10
  - 知道基本的数据转换
  - int() str() float() chr() ord()

基础篇

- Python函数参数是传值还是传址
  - Python中的函数通过引用传参
  - 明白传值和传址-两种不同的方式

基础篇

- 请问了解str和bytes之间的区别的么
  - bytes是一种包含8位值的序列，str是一种包含unicode字符的序列，不能用+等操作符混合操作
  - print(b‘知数堂’) 这样写对么

- 请简述isinstance和type的区别
  - type()不会认为子类是一种父类类型。
  - isinstance()会认为子类是一种父类类型。

基础篇

- Python中，lambda函数也叫匿名函数，及即没有具体名称的函数，它允许快速定义单行函数
- `lambda [arg1 [, agr2,.....argn]] : expression`

```
#单个参数的:
g = lambda x : x ** 2
print(g(2))
```

```
#多个参数的:
g = lambda x, y, z : (x + y) ** z
print(g(1,2,3)
```

基础篇

- 请问if表达式为假有哪几种情况
  - None False 0,0.0,0L '',(),[],{}

- 请问了解字符串的intern机制么

```
a = "abc"
b = "abc" #b=a
print(a is b) // True
a = ["abc"]
b = ["abc"]
print(a is b) // False
```

当创建a时，系统会首先创建一个PyStringObject对象出来，然后经过intern机制处理，接着查找经过intren处理过的PyStringObject对象，如果发现有该字符串对应的PyStringObject存在，则直接返回这个对象并且增加该字符串对象的引用计数，否则就把它加入到intern机制当中。由于a和b字符串值是一样的，所以当未b赋值时，直接就返回了a中的PyStringObject对象。所以他们的内存地址相同。

基础篇

- 1.举个例子，在**python命令行模式**下：为什么同样值a,b与c,d的结果却不一样呢？ -5,256
- >>> a = 1000
- >>> b = 1000
- >>> a is b False
- >>> c = 10
- >>> d = 10
- >>> c is d
- True

基础篇

- 请问A or B and C 是如何计算的
- 1+1*2
- 4 * 7 % 3
- A or B and C
- 4**3**2
- 1 & 2 & 3
- 4+1|2
- 4+4<<2

表 1 Python 运算符优先级和结合性一览表

运算符说明	Python运算符	优先级	结合性	优先级顺序
小括号	( )	19	无	
索引运算符	x[i] 或 x[i1: i2 [:i3]]	18	左	高
属性访问	x.attribute	17	左	^
乘方	**	16	左	|
按位取反	~	15	右	|
符号运算符	+（正号）、-（负号）	14	右	|
乘除	*、/、//、%	13	左	|
加减	+、-	12	左	|
位移	>>、<<	11	左	|
按位与	&	10	右	|
按位异或	^	9	左	|
按位或	|	8	左	|
比较运算符	==、!=、>、>=、<、<=	7	左	|
is 运算符	is、is not	6	左	|
in 运算符	in、not in	5	左	|
逻辑非	not	4	右	|
逻辑与	and	3	左	|
逻辑或	or	2	左	低
逗号运算符	exp1, exp2	1	左	

基础篇

- 请问了解生成器、迭代器么
  - 如果一个对象可以用于For循环，那么这个对象就是Iterable
  - 可以被next()函数调用并不断返回下一个值的对象称为迭代器：Iterator

```
for x in [1,2,3]:
    print(x)

for x in {"a": 1, "b": 2}:
    print(x)
```

我们可以说列表、词典都是Iterable

```
it = iter([1, 2, 3, 4, 5])
# 循环:
while True:
    try:
        # 获得下一个值:
        x = next(it)
    except StopIteration:
        # 遇到StopIteration就退出循环
        break
```

列表、词典不是Iterator，因为不能直接被next调用

基础篇

- 请同时对两个列表进行遍历—zip函数的使用

```
for x, y in zip(range(100), range(10)):
        print(x, y)
```

```
from itertools import zip_longest
for x, y in zip_longest(range(100), range(10)):
    print(x, y)
```

知道这段代码分别输出什么么

数据结构篇

- 请问以下代码会报错么

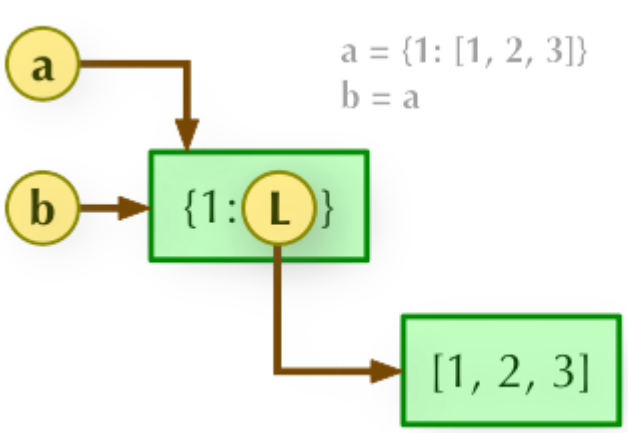
```
a = [1]
print(a[:20])
```

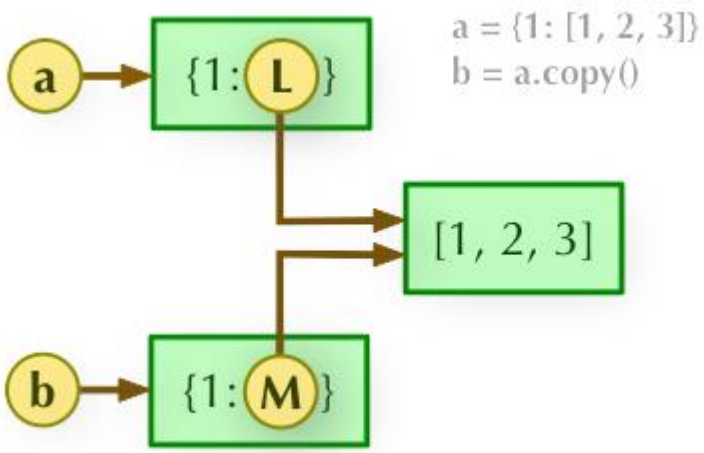
- 请写出以下代码的输出
- 切片在左边和右边的区别

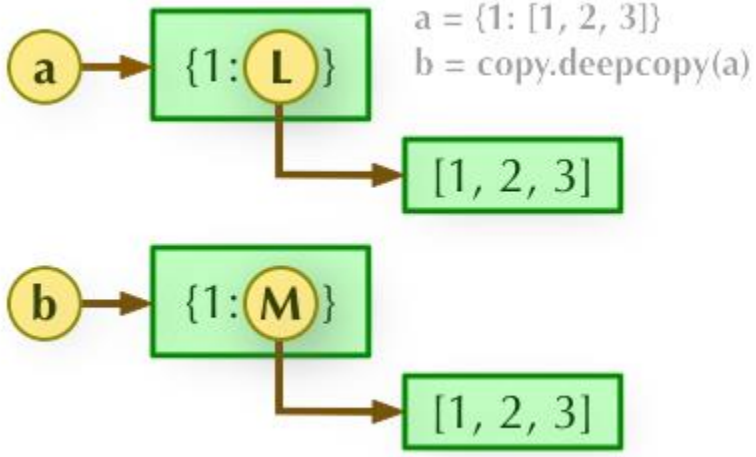
```
a = [1, 2, 3]
b = a
print('Before', a)
a[:] = [101,102, 103]
assert a is b
print('After ', a)
print('After ', b)
```

数据结构篇

- 请讲讲词典的直接赋值、浅拷贝、深拷贝
  - 直接赋值：其实就是对象的引用（别名）。
  - 浅拷贝(copy)：拷贝父对象，不会拷贝对象的内部的子对象。
  - 深拷贝(deepcopy)：copy 模块的 deepcopy 方法，完全拷贝了父对象及其子对象。







容器Collections

- Python附带一个Collections模块，包含了许多容器数据类型
  - Defaultdict
  - Counter
  - Deque
  - Namedtuple
  - Enum
  - OrderedDict

Defaultdict

- 该例子统计strings中某个单词出现的次数，并在counts字典中作记录。单词每出现一次，在counts相对应的键所存的值数字加1。但是事实上，运行这段代码会抛出KeyError异常，出现的时机是每个单词第一次统计的时候，因为Python的dict中不存在默认值的说法

```
strings = ('puppy', 'kitten',
'puppy', 'puppy',
           'weasel', 'puppy',
'kitten', 'puppy')
counts = {}

for kw in strings:
    counts[kw] += 1 # 会报错
```



```
trings = ('puppy', 'kitten', 'puppy',
'puppy',
           'weasel', 'puppy', 'kitten',
'puppy')
counts = {}

for kw in strings:
  if kw not in counts:
      counts[kw] = 1
  else:
      counts[kw] += 1
```

Defaultdict

```
# 我们用defaultdict改写前面一个例子
from collections import defaultdict

strings = ('puppy', 'kitten', 'puppy', 'puppy',
           'weasel', 'puppy', 'kitten', 'puppy')
counts = defaultdict(lambda: 0)  # 使用lambda来定义简单的函数

for s in strings:
    counts[s] += 1
```

Counter

- 支持便捷和快速地计数。一个Counter是dict子类，用于计数可哈希的对象。这是一个无序的容器，元素被作为字典的key存储，它们的计数作为字典的value存储。Counts允许是任何整数，包括0和负数。

```
from collections  import Counter
c = Counter()
for ch in 'programing':
    c[ch] = c[ch] +1
print(c)
#Counter({'r': 2, 'g': 2, 'p': 1, 'o': 1, 'a': 1, 'm': 1, 'i': 1, 'n': 1})
```

Deque

- deque提供了一个双端队列，可以从头/尾两端添加或者删除元素

```
from colletions import deque
d = deque() # 创建对象
d.append('1')
d.append('2')

d.popleft() # 左边取出数据
d.pop() # 右边取出数据

# 限制列表大小
d = deque(maxlen=30)

#任意一端扩展数据
d.extendleft([0])
d.extend([6,7,8])
```

Namedtuple

- 针对简单任务的容器，不必使用整数索引来访问namedtuples的数据，可以像词典一样访问namedtuples，但是namedtuples是不可变的

```
from collections import namedtuple
Point = namedtuple('Point', ['x', 'y'])
# Animal = namedtuple('Animal', 'name age type') 这种方法也可以
p = Point(1, 2)
```

Enum

- 当我们需要定义常量时，可以使用枚举类

```
from enum import Enum
from collections import namedtuple
class DbType(Enum):
    mysql = 1
    pg = 2
    oracle = 3

Instance = namedtuple("db", "host port type")
m1 = Instance(host="127.0.0.1", port=3306, type=DbType.mysql)
print(m1)
```

OrderedDict

- 使用dict时，Key是无序的。在对dict做迭代时，我们无法确定Key的顺序。
- 如果要保持Key的顺序，可以用OrderedDict：

```
>>> from collections import OrderedDict
>>> d = dict([('a', 1), ('b', 2), ('c', 3)])
>>> d # dict的Key是无序的
{'a': 1, 'c': 3, 'b': 2}
>>> od = OrderedDict([('a', 1), ('b', 2), ('c', 3)])
>>> od # OrderedDict的Key是有序的
OrderedDict([('a', 1), ('b', 2), ('c', 3)])
```