一 课程回顾

```
# 1.给定两个字符串，使用拼接符将它们连接起来。
# str_1 = "Hello"
# str_2 = "World"
# print(str_1+' '+str_2)


# 2.使用占位符将一个姓名和年龄拼接成一句话。
# name = "小明"
# age = 18
# a = '我的名字是{},我今年{}岁'.format(name,age)
# print(a)

# 3.使用 join 方法将一组字符串按照指定的字符进行连接。(I-love-Python)
# words = ["I", "love", "Python"]
# result = '-'.join(words)
# print(result)

# 4.使用 format 方法将三个字符串按照指定的顺序进行连接。(他真帅是吗？)
# str_1 = "他"
# str_2 = "真帅"
# str_3 = "是吗？"
# b = "{}{}{}".format(str_1,str_2,str_3)
# print(b)

# 5.使用 format 方法根据下标位置将三个字符串进行连接。(他真帅是吗？)
# str_1 = "他"
# str_2 = "真帅"
# str_3 = "是吗？"
# c = "{0}{1}{2}".format(str_1,str_2,str_3)
# print(c)

# 6.使用 format 方法通过给变量重新命名进行字符串连接。(他真帅是吗？)
# str_1 = "他"
# str_2 = "真帅"
# str_3 = "是吗？"
# d = "{first}{qqq}{ccc}".format(first = str_1,qqq = str_2, ccc = str_3)
# print(d)


# 7.使用 f-format 直接将三个字符串进行连接。
# str_1 = "他"
# str_2 = "真帅"
# str_3 = "是吗？"
# e = f'{str_1}{str_2}{str_3}'
# print(e)



# 11.将一个字符串中所有的字符转换为大写。
# data_string = "hello world"
```

```
# f = data_string.upper()
# print(f)


# 13

# 假设有一个字符串 "Hello, my name is John. I am 25 years old."，请完成以下操作(years old--岁的意思)：  (不能直接使用下标  要用字符串的方法来写)

#   1.使用索引和切片获取并输出该字符串中的姓名（John）。
# data_string = "Hello. my name is John. I am 25 years old."
# a = data_string.index('is ')+3  # 开始值  j
# b = data_string.index('.',a)  # 结束值
# c = data_string[a:b]
# print(c)


#   2.使用整数计算获取并输出该字符串中年龄的两倍。
# data_string = " years Hello, my name is John. I am 25 years old."
# a = data_string.index('am ')+3  # 开始值 2
# b = data_string.index(' years',a)  # 结束值
# c = int(data_string[a:b])
# d = c * 2
# print(d)



#     3.将该字符串中的名字（John）替换为另一个名字，并输出替换后的字符串。
# data_string = "Hello, my name is John. I am 25 years old."
# new_name = 'zuoshou'
# a = data_string.replace('John',new_name)
# print(a)


#     4.使用字符串方法，将该字符串中的年龄替换为当前年份（2023）减去出生年份（出生年份为1980），并输出替换后的字符串。
a = 1980  # 当前年份
b = 2023 # 出生年份
data_string = "Hello, my name is John. I am 1980"  # 2023 - 1980
c = b - a
q = data_string.find('am ')  #   q = a的下标
w = data_string.replace(data_string[q+3:q+7],str(c))
print(w)
```

上节课没有讲完的知识点

字典的key可以是任意的不可变类型

字典相当于是索引不可变类型的列表， 列表则相当于key 只能是整数的字典

总结: 字典的索引就是key

字典允许直接对不存在的key赋值，这样就增加一个键值对

```
print(dir(dict))


['__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__',
'__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__',
'__getitem__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__ior__',
'__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__',
'__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__ror__',
'__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__',
'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem',
'setdefault', 'update', 'values']

主要讲下面比较重要的方法
 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem',
'setdefault', 'update', 'values']
```

clear()

用于清空字典中所有的键值对

```
cars = {'BMW':88,'benchi':83,'aodi':95}
print(cars)
cars.clear()
print(cars)
```

get()

```
cars = {'BMW':88,'benchi':83,'aodi':95}
print(cars.get('BMW'))
print(cars.get('baoshijie'))
# print(cars['baoshijie'])
```

update()

可以使用一个字典所包含的键值对来更新已经有的字典，如果被更新的字典包含对应的键值对，那么原来的值会被覆盖，如果不包含，就会被添加进去

```
cars = {'BMW':88,'benchi':83,'aodi':95}
cars.update({'BMW':4.5,'baoshijie':9.3})
print(cars)
```

items(),keys(),values()

dict_items,dict_keys,dict_values对象，python不希望用户直接操作这几个方法，但是可以通过list函数把他们转成列表

```
items ==> 取键值对


cars = {'BMW':88,'benchi':83,'aodi':95}
ims = cars.items()
print(ims)
print(type(ims))
print(list(ims))
print(list(ims)[1])
```

```
keys() ==> 取键

cars = {'BMW':88,'benchi':83,'aodi':95}
ims = cars.keys()
print(ims)
print(type(ims))
print(list(ims))
print(list(ims)[1])
```

```
values() ==> 取值

cars = {'BMW':88,'benchi':83,'aodi':95}
ims = cars.values()
print(ims)
print(type(ims))
print(list(ims))
print(list(ims)[1])
```

pop()

用于获取指定的key对应的value，并且删除这个键值对

```
cars = {'BMW':88,'benchi':83,'aodi':95}
print(cars.pop('benchi'))
print(cars)
```

setdefault()

能返回指定的key对应的value值，如果键值对不存在，就先为这个键赋值，然后返回这个键对应的值

```
cars = {'BMW':88,'benchi':83,'aodi':95}
print(cars.setdefault('baoshijie',92)) # 两个参数  第一个参数 key的名字， 第二个参数设置key对应value 如果不存在就会新增键值对
print(cars)

原有key的value值 不会被覆盖
```

二 数据类型进阶(三)

2.1 公共功能

长度

```
info = {'age':12,'xingming':'zuoshou','shengao':175.0}
data = len(info)
print(data)
```

for循环

```
info = {'age':12,'xingming':'zuoshou','shengao':175.0}
# 直接遍历这个字典，可以获取所有的键
# for i in info:
#     print(i)
#
# for i in info.keys():
#     print(i)

# for i in info.values():
#     print(i)

# for key in info.items():
#     print(key)
```

2.2 集合 set

元素都是唯一的，互不相同的

```
语法:
    {name1,name2,....namen}
集合名 = {数据1，数据2，。。。。数据n}# 如果有重复数据，是不会被添加到内存空间的
```

无法存储列表，集合，字典这些数据类型，否则报错

```
# print({{'a':1}})
# print({[1,2,3]})
# print({{1,2,3}})
```

要注意的是:集合中的数据必须保证是唯一的，集合对于每种数据元素，只会保留一份，也就是去重

```
s = {1,2,1,(1,2,3),'c','c'}
print(s)
```

```
set1 = {1,2,3,4,4,5,5,1,1,2,3,8,7,9,4,5,6,7,8,9,7,8,9}
print(set1)  # 正常一个集合，里面的数据是默认去重的
# print(set1[1])  # 报错
print(type(set1))
```

注意: 创建空集合的时候必须使用set()而不能是{}，因为{}默认是空字典

```
li1 = []
tu1 = ()
s1 = ''
dict1 = {}
set1 = set()
print(type(li1))
print(type(tu1))
print(type(s1))
print(type(dict1))
print(type(set1))
```

集合基本操作 ==》 去重

```
li = [1,2,3,4,5,6,7,8,9,4,1,1,1,1]
li1 = []
for i in li:  # 遍历li内部所有的元素
    if i not in li1:
        li1.append(i)
print(li1)

print(list(set(li)))
```

运算符操作	python运算符	含义
交集	&	取两集合公共的元素
并集	|	取两集合全部的元素
差集	-	取一个集合中另一集合没有的元素
成员运算	in 和 not in	判断一个某个元素是否在或者不在集合中

```
set1 = {1,2,3} # set1 - set2 set1中有数据在set2当中  就会将那份数据删掉
set2 = {3,4,5}
print(set1 & set2)  # 交集  3  获得公共部分
print(set1 | set2)  # 并集  {1, 2, 3, 4, 5}  获得并集
print(set1 - set2)  # 差集  取一个集合中另一个集合没有的元素，将set中属于set2的元素删除
print(set2 - set1)
print(3 in set2)
```

集合的作用:

1. 存储非重要数据
2. 用于将序列类型去重，逻辑判断

2.2.1 增

```
add() 参数为要添加的对象，通过多次添加数据可以发现添加后的元素位置不确定

语法: 集合名.add(元素)  ==》 无顺序

s = {'左手'}
s.add('汪秋')
```

```
print(s)
s.add('学家')
print(s)
s.add('郭靖')
print(s)

update() 参数为序列类型，会将每一个元素迭代添加到序列中(随机添加)
s = {'左手'}
s.update('123')
print(s)
s.update([4,5,6])
print(s)
```

2.2.2 删

```
pop() 随机删除一个元素
实际上在进行代码实验的时候并不是随机的
仅仅是在集合元素是字符串类型时，并且在cmd运行的时候才会随机删除，pycharm默认保持删除第一个元素
s = {'左手'}
s.update('123')
print(s)
s.update([4,5,6])
print(s)
s.pop()
print(s)
s.pop()
print(s)


remove() 有参数，参数就是要删除的元素，如果元素不存在就会报错
s = {'左手'}
s.update('123')
print(s)
s.update([4,5,6])
print(s)
s.remove('1')
print(s)
s.remove('2')
print(s)
s.remove('8')
print(s)


discard() 跟remove类似，但是元素不存在也不会报错
s = {'左手'}
s.update('123')
print(s)
s.update([4,5,6])
print(s)
s.remove('1')
print(s)
s.remove('2')
print(s)
s.discard('8')
print(s)
```

```
clear()  清空集合中的元素
s = {'左手'}
s.update('123')
print(s)
s.update([4,5,6])
print(s)
s.clear()
print(s)


del 集合名() 删除集合

s = {'左手'}
s.update('123')
print(s)
s.update([4,5,6])
print(s)
del s
print(s)  # 整个把变量删除了， 所以会报错，先删除了变量，再去打印s就会报错
```

2.2.3 改

```
由于set中的数据没有索引，没办法去定位一个元素，所以没办法直接修改
先删除再添加
s = {'A','B','C','D'}
# 把A改为E
s.remove('A')
s.add('E')
print(s)

集合是没有改的方法
```

2.2.4 查

```
set是一个可迭代对象，可以通过for循环进行遍历查询
s = {'A','B','C','D'}
for i in s:
    print(i)
```

作业

1. 创建一个空字典my_dict。

2. 向字典my_dict中添加键值对'apple': 3、'banana': 6和'orange': 4。

3. 获取字典my_dict中键为'banana'的值，并将其存储在变量value中。

4. 将字典my_dict中键为'orange'的值增加2。

5. 删除字典my_dict中键为'apple'的键值对。

6. 检查字典my_dict中是否存在键'grape'。

7. 获取字典my_dict中所有键，并将其存储在列表keys中。

8. 获取字典my_dict中所有值，并将其存储在列表values中。

1.创建一个空集合my_set。
2.向集合my_set中添加元素'apple'、'banana'和'orange'。
3.从集合my_set中删除元素'banana'。
4.创建另一个集合my_set，包含元素'kiwi'和'grape'。
5.获取两个集合my_set和my_set2的并集。
6.获取两个集合my_set和my_set2的交集。
7.获取两个集合my_set和my_set2的差集。
8.判断集合`my_set`是否是集合`my_set2`的子集。