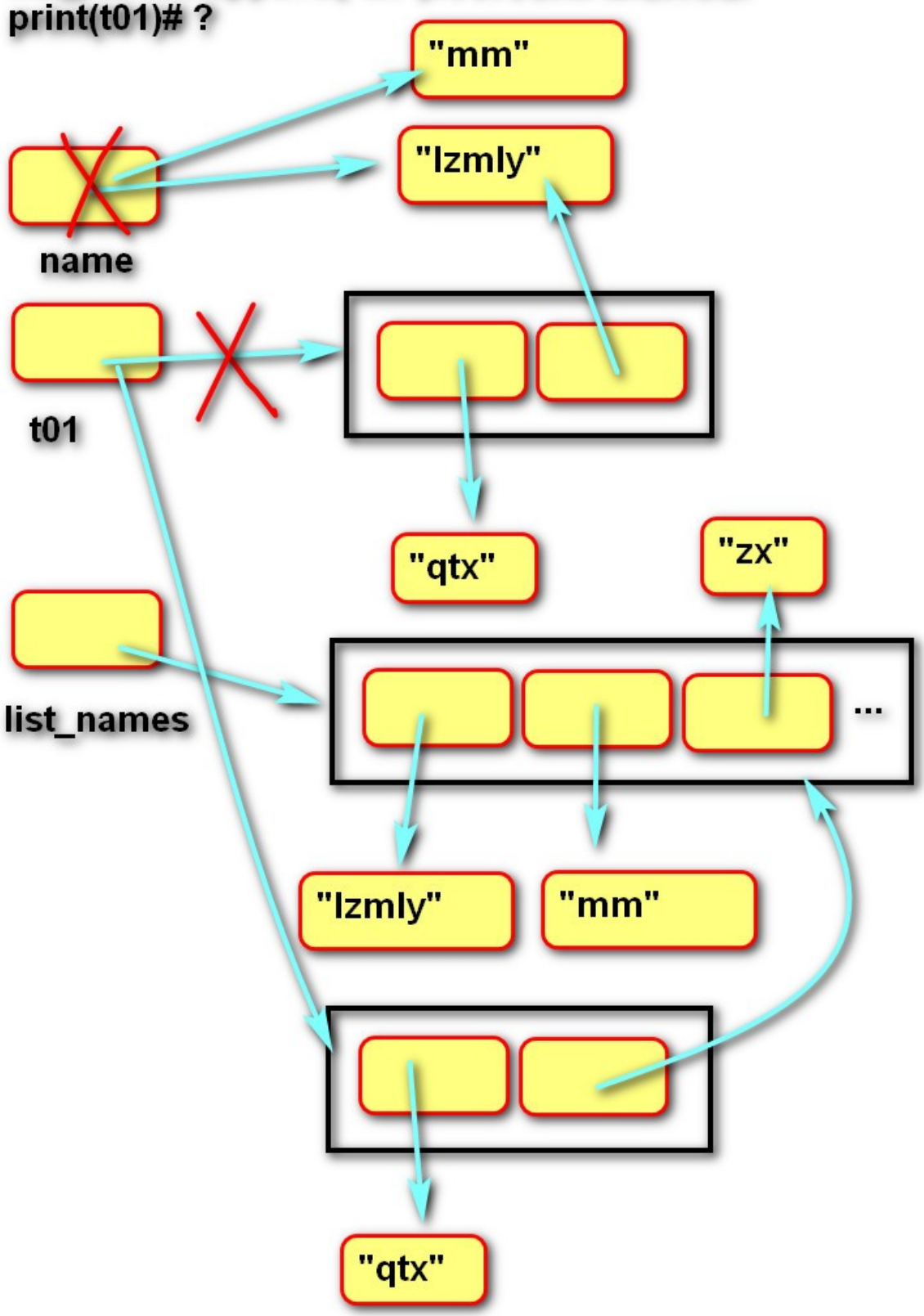
```
name = "lzmly"
t01 = ("qtx",name)# 将变量存储的数据赋值给元组第二个元
name = "mm"# 修改变量存储的地址
print(t01)#?

list_names = ["lzmly","mm"]
t01 = ("qtx",list_names)
list_names.append("zx")#向列表添加新元素
print(t01)# ?
```



集合 set

定义

1. 由一系列不重复的不可变类型变量组成的可变散列容器。
2. 相当于只有键没有值的字典(键则是集合的数据)。

基础操作

1. 创建空集合：
   集合名 = set()
   集合名 = set(可迭代对象)
2. 创建具有默认值集合：
   集合名 = {1, 2, 3}
   集合名 = set(可迭代对象)
3. 添加元素：
   集合名.add(元素)
4. 删除元素：
   集合名.discard(元素)

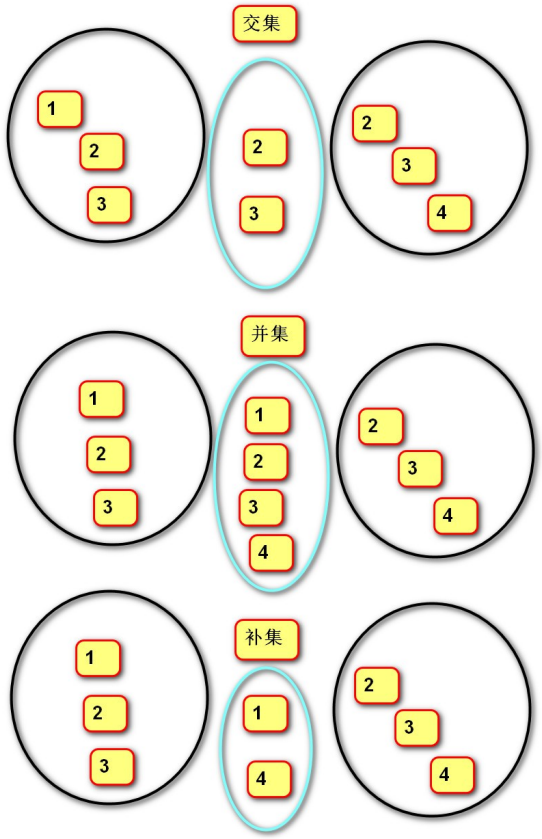
运算

1. 交集&：返回共同元素。
   s1 = {1, 2, 3}
   s2 = {2, 3, 4}
   s3 = s1 & s2 # {2, 3}

2. 并集：返回不重复元素
   s1 = {1, 2, 3}
   s2 = {2, 3, 4}
   s3 = s1 | s2 # {1, 2, 3, 4}

3. 补集-：返回只属于其中之一的元素
   s1 = {1, 2, 3}
   s2 = {2, 3, 4}
   s1 - s2 # {1} 属于 s1 但不属于 s2

   补集^：返回不同的的元素
   s1 = {1, 2, 3}
   s2 = {2, 3, 4}
   s3 = s1 ^ s2 # {1, 4} 等同于(s1-s2 | s2-s1)



4. 子集<：判断一个集合的所有元素是否完全在另一个集合中
5. 超集>：判断一个集合是否具有另一个集合的所有元素

```
s1 = {1, 2, 3}
s2 = {2, 3}
s2 < s1  # True
s1 > s2  # True
```

6. 相同或不同== !=：判断集合中的所有元素是否和另一个集合相同。

```
s1 = {1, 2, 3}
s2 = {3, 2, 1}
s1 == s2  # True
s1 != s2  # False
子集或相同,超集或相同 <=  >=
```

```
"""
    集合
        不能重复   --->  将其他具有重复元素的容器转换为集合(去重复)
        数学运算   --->
    练习:exercise01.py
"""
# 创建空集合
s01 = set()
print(type(s01))
# 创建具有默认值的集合
```

```python
s02 = {"a","b"}
s02 = set("abcabc")#
print(s02)
tuple02 = tuple(s02)
# 增加
s02.add("qtx")
s02.add("mly")
# 删除
s02.remove("mly")# 如果没有该元素 则报错
print(s02)
# 获取所有
for item in s02:
    print(item)
# 数学运算
# 交集
s1 = {1, 2, 3}
s2 = {2, 3, 4}
s3 = s1 & s2  # {2, 3}
print(s3)
# 并集
s1 = {1, 2, 3}
s2 = {2, 3, 4}
s3 = s1 | s2  # {1, 2, 3, 4}
print(s3)
# 对称补集
s1 = {1, 2, 3}
s2 = {2, 3, 4}
s3 = s1 ^ s2  # {1, 4}  等同于(s1-s2 | s2-s1)
print(s3)
# 补集
print(s2 - s1)# {4}
print(s1 - s2)# {1}
# 子集超集
s1 = {1, 2, 3}
s2 = {2, 3}
print(s2 < s1)  # True
print(s1 > s2)  # True
```

集合推导式

1. 定义：
   使用简易方法，将可迭代对象转换为集合。
2. 语法:
   {表达式 for 变量 in 可迭代对象}
   {表达式 for 变量 in 可迭代对象 if 条件}

"""

```
"""
# 外层循环做1次，内层循环做3次
# 外层循环控制行
# 内层循环控制列
for r in range(2):#     0       1
    for c in range(3):#012     012
        print("*",end =" ")# 一行打印
    print()# 换行
```

"""

```
    列表推导式嵌套
    练习:exercise08.py
"""
list01 = ["a","b"]
list02 = ["A","B","C"]
list_result = []
# list01[0]  +  list02[0]
# list01[0]  +  list02[1]
# list01[0]  +  list02[2]
# for c in range(3):
#     list_result.append(list01[0] + list02[c])
#
# for c in range(3):
#     list_result.append(list01[1] + list02[c])
# for r in range(2):
#     for c in range(3):
#         list_result.append(list01[r] + list02[c])
for r in list01:
    for c in list02:
        list_result.append(r + c)
list_result = [r + c for r in list01 for c in list02]
print(list_result)
```

固定集合 frozenset

定义

不可变的集合。

作用

固定集合可以作为字典的键,还可以作为集合的值。

基础操作

创建固定集合：frozenset(可迭代对象)

运算

等同于 set

```
"""
    固定集合 frozenset
        主要作用：与其他容器互相转换
        次要作用:可以作为字典的 key
"""
# 创建
s01 = frozenset(["a","a","b"])
print(s01)
```

函数 function

pycharm 相关设置

1. “代码自动完成”时间延时设置
   File -> Settings -> Editor -> General -> Code Completion -> Autopopup in (ms):0
2. 快捷键：
   Ctrl + P　　参数信息（在方法中调用参数）
   Ctrl + Q　　快速查看文档
   Ctrl + Alt + M　　提取方法

定义

1. 用于封装一个特定的功能，表示一个功能或者行为。
2. 函数是可以重复执行的语句块, 可以重复调用。

作用

提高代码的可重用性和可维护性（代码层次结构更清晰）。

定义函数

1. 语法：

   def 函数名(形式参数):

   函数体

2. 说明：

   def 关键字：全称是 define，意为”定义”。

   函数名：对函数体中语句的描述，规则与变量名相同。

   形式参数：方法定义者要求调用者提供的信息。

   函数体：完成该功能的语句。

3. 函数的第一行语句建议使用文档字符串描述函数的功能与参数。

```
"""
    函数：表示一个功能
    参数：用功能时提供的信息。
    练习：exercise09.py
    练习：exercise10.py
    17:05 上课
"""
# 做功能(参数)
def attack(count):
    """
        攻击
    :param count: 整数类型的攻击次数
    """
    for i in range(count):
        print("勾拳")
        print("直拳")
        print("天马流星拳拳")
        print("临门一脚")
#.....
# print("直拳")
# print("勾拳")
# print("天马流星拳拳")
# print("临门一脚")
# 用功能
attack(1)
# print("直拳")
# print("勾拳")
# print("天马流星拳拳")
# print("临门一脚")
attack(2)
```

调用函数

1. 语法：函数名(实际参数)
2. 说明：根据形参传递内容。

返回值

1. 定义：
   方法定义者告诉调用者的结果。
2. 语法：
   return 数据
3. 说明：
   return 后没有语句，相当于返回 None。
   函数体没有 return，相当于返回 None。

"""

```
    函数：【一个】功能
    返回值：做函数的人，给用函数的人的结果
    练习：exercise11.py
    练习：exercise12.py
"""
# 小李同学
# 定义两个数字相加的函数
def add(number_one, number_two):
    # 逻辑处理
    return number_one + number_two  # 返回结果，退出方法
    print("看见我了吗？")  # return 以后的语句不在执行
# 获取数据
one = int(input("请输入第一个整数："))
two = int(input("请输入第二个整数："))
re = add(one, two)
# 显示结果
print("结果是：" + str(re))
```