python总结

前端

```
Angular
React
Vue
```

```
Bootstrap 是全球最受欢迎的前端组件库，用于开发响应式布局、移动设备优先的 WEB 项目。
```

PEP

PEP8

Python 代码风格指南

https://www.python.org/dev/peps/pep-0008/

PEP484

类型提示

https://www.python.org/dev/peps/pep-0484/

PEP526

变量注释的语法

https://www.python.org/dev/peps/pep-0526/

PEP20

The Zen of Python

https://www.python.org/dev/peps/pep-0020/

总结

1、一点点拿，一点点算，提高效率

```
# 1. 将一个文件拆分为两个小文件,按照字节数平均拆分， 使用父进程和子进程同时进行
# 一个进程获取半部分,换一个进程获取下半部分
```

2、宏观与微观

3、选择语句两种思维

4、python帮助

```
help()
dir()
object.__doc__
```

5、索引

```
索引：取索引就是在查找下一个内存空间
```

复习

```
"""
  复习
  \1. python：免费，开源，跨平台，动态，面向对象的编程语言
  \2. 执行方式：交互式
              文件式
  \3. 执行过程：源代码-编译->字节码-解释->机器码
        |-----1次-----|-----每次---|
  \4. 学习方法:知识点必须理解(定义 / 作用 / 适用性 / 语法)
            整理笔记(三合一)
            当天练习必须独立完成
"""

"""
  day02 复习
  数据基本运算
    变量：关联一个对象的标识符
      变量名 = ?
        变量没有类型
    数据类型:
      None
      int    1    2
      float     1.0  2.5
      str    ""   "字符"
      bool     True  False
      复数  complex

•    类型转换
•      int(数据)    float(数据)
•      str(数据)    bool(数据)
•      如果数据的格式不正确，会错误。
•        例如：int("100+")
•      如果数据表示"没有",转换结果为F a l s e
•        bool(1) --> True
•        bool("") -->False

•    运算符
•      算数运算符：+  -  *    /  / /  %  **
•      增强运算符:+=  -=  *=    /=  / /=  %=  **=
•        a = 10
•        a = a + 5
•        a += 5
•      比较运算符:>  <  >=  <=  ==  !=
•      逻辑运算符: 1 > 2  "a" == "b"
•            False or False
•          与  and : 一假俱假
```

```
•           或  or :一真俱真
"""

a = 1
a = "a"
a = True

\# 问题：控制台中会出现什么
\# 短路逻辑：逻辑运算时，尽量将复杂(耗时)的判断放在后边。
num = 1
\# and 发现False，就有了结论,后续条件不再判断。
\# re = num > 1 and input("请输入：") == "a"

\# or 发现True，就有了结论,后续条件不再判断。
re = num + 1 > 1 or input("请输入：") == "a"


"""
  day03 复习
  语句
    选择语句
      if bool类型的条件:
        满足条件执行的语句
      else:
        不满足条件执行的语句

•      if 条件1:
•        满足条件1执行的语句
•      if 条件2:
•        满足条件2执行的语句
•      if 条件3:
•        满足条件3执行的语句

•      if 条件1:
•        满足条件1执行的语句
•      elif 条件2:
•        不满足条件1，满足条件2执行的语句
•      elif 条件3:
•        不满足条件1/2,满足条件3执行的语句
•      else:
•        以上条件都不满足执行的语句

•    循环语句
•      if 条件:
•         满足条件执行一次
•      else:
•        不满足条件执行一次

•      while 条件:
•        满足条件一直执行
•      else:
•        不满足条件执行一次

•    跳转语句
•      break
"""
```

```
"""
  day04 复习
  语句
    循环语句
      for  + range()：固定次数的循环
      while:根据条件执行的循环

•      range(开始,结束,步长)
•        range(2,6,2)->2 4
•        range(2)->0 1
•        range(2,2)->

  容器
    字符串str:不可变    编码值  utf－8
      字面值
        单引号  双引号  三引号（所见即所得）
        转义符 \字符
        字符串格式化
        "..."+变量1+“.."+变量2+".."
        "...%s..%f.."%(变量1,变量2)

•    通用操作
•      数学运算符 +  *
•      成员运算符  元素 in 容器
•      索引:定位单个元素
•      切片：定位多个元素
•      函数:len(容器) 长度
"""
name = '大强'
name = "小强"
print(name)#小强




"""
  day05 复习
  容器
    通用操作
    字符串:不可变    存储编码值    序列
    列表:可变    存储变量          序列
      基础操作
        1.创建:[数据]  list(容器)
        2.定位：索引    切片
          \# 从列表中获取一片元素组成新列表
          变量 = 列表名[切片]
          \# 修改一片元素
          列表名[切片] = 变量
        3.删除:
          del 列表名[索引/切片]
          列表名.remove(元素)
          从列表中删除多个元素,建议倒序删除.
        4.增加:
          列表名.append(元素)
```

```
          列表名.insert(索引,元素)
        \5. 遍历所有元素
          下列代码
"""
\# 遍历所有元素
list01 = [3, 4, 4, 5, 6]
\# 打印列表
\# print(list01)
\# 正向
for item in list01:
  print(item)

\# 反向(索引)
\# 3  2  1  0
for i in range(len(list01) - 1, -1, -1):
  print(list01[i])

\# -1 -2 -3 -4
for i in range(-1, -len(list01) - 1, -1):
  print(list01[i])









"""
  day06 复习
  容器
    字符串:不可变 存储编码值 序列
    列表:可变  存储变量 序列
      预留空间
      扩容: 开辟更大的空间
          拷贝原有数据
          替换引用
    元组:不可变  存储变量  序列
      按需分配
    字典:可变 存储键值对 散列
    集合:可变 存储键 散列
    固定集合:不可变 存储键 散列
"""
list01 = []
list01 = ["qtx", "xz", "jd"]
list01.append("mm")
list01.insert(1, "wt")

\# item 变量指向列表中的元素
for item in list01:
  print(item)

\# 变量 i表示索引
for i in range(len(list01)):
  print(i)

\# 修改
```

```
list01[0] = "QTX"

\# 删除
list01.remove("mm")

dict01 = {"qtx": 100, "xz": 65, "jd": 85}
dict01["mm"] = 95
\# 获取所有元素
for key in dict01:
  print(key)
  print(dict01[key])

for value in dict01.values():
  print(value)

for key, value in dict01.items():
  print(key)
  print(value)

\# 修改
dict01["qtx"] = 101

\# 删除
del dict01["mm"]

list02 = ["看书", "编程", "美食"]
dict02 = {"qtx": list02}
list02.append("听音乐")
print(dict02)



"""
  day07 复习
  能力提升for for
    \# 结论：外层循环执行一次，内层循环执行多次。
           外层控制行，内层控制列.

•    for r in range(2):#   0   1
•      for c in range(3):#012  012
•        pass

  函数
    定义:功能，使用一个名称，包装多个语句。
    语法:
      做
        def 名字(形参):
          函数体

•      用
•        名字(实参)
"""

list01 = [23,34,4,6]
for r in range(len(list01) - 1):
  \# 作比较
  for c in range(r + 1, len(list01)):
    \# list01[2]  list01[c]
```

```
    if list01[r] > list01[c]:
      list01[r], list01[c] = list01[c], list01[r]



"""
  day08 复习
  函数
    基础语法
      定义函数
        def 函数名称(形参):
          函数体


      调用函数
        函数名称(实参)

•    基础概念
•      参数:调用者传递给定义者的信息.
•        定义者要求调用者必须提供的信息.

•      返回值:定义者传递给调用者的结果

•    参数
•      实际参数
•        位置实参:实参与形参按位置对应
•          序列实参：参数过多，可以将实参存储在序列中.
•              用星号拆分后与形参对应.

•        关键字实参：实参与形参按名字对应
•          字典实参：参数过多，可以将实参存储在字典中.
•              用双星号拆分后与形参对应.

•      形式参数
•        默认形参：给形参提供一个默认值，实参可以不提供.
•        位置形参
•          星号元组形参：让位置实参个数无限
•        命名关键字形参：要求实参必须是关键字实参
•          双星号元组形参:让关键字实参个数无限
"""


def fun01(a, b, c):
  print(a)
  print(b)
  print(c)


\# 位置实参
fun01(1, 2, 3)
list01 = [1, 2, 3]
\#   星号:拆分序列
fun01(*list01)

dict01 = {"a": 1, "c": 3, "b": 2}
\#   双星号:拆分字典
fun01(**dict01)

```

```
336
337 def fun02(a=0, b=0, c=0):
338   print(a)
339   print(b)
340   print(c)
341
342 \# 关键字实参
343 fun02(c=3)
344
345
346 \# 将实参合并为一个元组
347 def fun03(*args):
348   print(args)
349
350
351 fun03(2, 3, 4, 4, 5)
352
353
354 \# 将实参合并为一个字典
355 def fun03(**kwargs):
356   print(kwargs)
357
358
359 fun03(a=1, b=2)
360
361
362 def fun04(*, name):
363   print(name)
364
365
366 fun04(name="")
367
368 print("#", "*", 123,sep="--",end=" ")
369 print("#", "*", 123,"--"," ")
```

```
1  # is与==的区别，及整数池
2  # a = [1, 2]
3  # b = [1, 2]
4  # print(a is b)
5  # print(a == b)
6  # print(id(a[0]))
7  # print(id(b[0]))
8
9  # 生成器，创建列表
10 # number = list(range(10))
11
12 # 索引：正序与倒序一起用
13 # list01 = [1, 2, 3, 4, 5]
14 # print(list01[3: -4: -1])
15
16 # 特殊情况
17 # for i in range(1):
18 #     print(True)
19 # print(False)
```

```
# import keyword
# print(keyword.kwlist)
# ['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue',
# 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global',
# 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise',
# 'return', 'try', 'while', 'with', 'yield'
```

```
print(bool("".encode()))
print(len("".encode()))
print("已存入".encode())
print("".encode())
```

调试方法

```
dir()
print()
type()
id()
len()
__dict__
time()
```

str

```
# 返回 str 在 string 里面出现的次数，如果 beg 或者 end 指定则返回指定范围内 str 出现的次数
count(str, beg= 0,end=len(string))
# 返回"标题化"的字符串,就是说所有单词都是以大写开始，其余字母均为小写(见 istitle())
title()
# 特殊写法
# a = list(("aaa"))
# print(a)  # ['a', 'a', 'a']
# b = ("aaa")
# print(type(b))  # <class 'str'>

print("小明""今年", 20, "岁")
print("小明" "今年" "岁")
print(len("小明" "今年" "岁"))
```

list -->str

```
"""
    list  -->  str

    字符串 = "连接符".join(列表) #连接符可以为空。
"""
list01 = [1,"1",2, "2"]
list02 = ["1",1,2, "2"]
result1 = "".join(list01)
print(result1)
# TypeError: sequence item 0: expected str instance, int found
```

```
11
12 result2 = "".join(list02)
13 print(result2)
14 # TypeError: sequence item 1: expected str instance, int found
15
```

str --->list

```
1  """
2      str -->  list
3
4      列表 = “a-b-c-d”.split(“分隔符”)
5  """
6  # 需求：将一个字符串描述的多个信息分别提取出来(列表)
7  names = "齐天大圣-猪八戒-唐僧"
8  list_names = names.split("-")
9  for item in list_names:
10     print(item)
```

list

```
1  list1 = [1, 2, 3, 4, 5]
2  for i in list1:
3      print(list1.pop(0), end=' ')
4  # result:
5  # 1 2 3
6
7  list1 = [1, 2, 3, 4, 5]
8  for i in range(n:= len(list1)):
9      print(list1.pop(0), end=' ')
10 print()
11 print(list1)
12 # result:
13 # 1 2 3 4 5
14 # []
15
16 list1 = [1, 2, 3, 4, 5]
17 list2 = []
18 for i in range(n:= len(list1)):
19     e = list1.pop(0)
20     list2.append(e)
21     print(e, end=' ')
22 print()
23 print(list1)
24 print(list2)
25 # result:
26 # 1 2 3 4 5
27 # []
28 # [1, 2, 3, 4, 5]
29
30 list1 = [0 for __ in range(5)]
31 print(list1)
32
33 dict1 = {'a': 1, 'd': 3, 'w': 5, 't': 9, 'v': 6}
34 # for key in sorted(dict1, key=lambda k: dict1[k], reverse=True):
```

```
#     print(key)
re = sorted(dict1, key=lambda k: dict1[k], reverse=True)
print(re)
# [0, 0, 0, 0, 0]
# ['t', 'v', 'w', 'd', 'a']
```