

Python 复习知识点

知识点

- 1. Python程序需要**描述数据**和**操作数据**。
- 2. Python程序区分大小写
- 3. 标识符的基本要求 - 驼峰法则(camelCase) / 匈牙利法则(Hungarian notation)
 - iPhone
 - eBay
 - johnSmith
- 4. 变量赋值及相应的类型(主要的基本数据类型)

类型	典型值
布尔型(bool)	True , False
整形(int)	1 , 100
浮点型(float)	3.1415926
复数(complex)	1+2j
字符串(string)	'hello'

- 5. 注意 = 与 == 的区别
 - 赋值运算符(=) - a = 10 把 10 赋给变量 a 。
 - 比较运算符(==) - a == 10 比较 a 和 10 是否相等，相等返回 True ，不相等返回 False 。
- 6. 特殊字符换行符 \n

In [1]:

```
print("hello\nworld")
```

hello
world

In [2]:

```
print("hello", end="\n\n")
```

hello

In [3]:

```
print('a', 'b', 'c', sep='$$')
```

a\$\$b\$\$c

In [4]:

```
print(True, 1, 3.14, 1+2j, 'hello', sep=', ')
```

True, 1, 3.14, (1+2j), hello

- 7. Python语法的缩进格式(严格的逻辑关系、语句块)

Guido van Rossum 认为使用缩进进行分组非常优雅，并且对提高普通 Python 程序的清晰度有很大贡献。大多数人会在一段时间后学会喜欢这个功能。

In [5]:

```
a, b = 0, 1
while a < 10:
    print(a)
    a, b = b, a+b
```

0
1
1
2
3
5
8

- 8. 输入内置函数 input() 的使用、返回值。

In [6]:

```
s = input('输入你的年龄:')
print(s)
```

20

- 9. 格式化输出 print() 的应用，包括宽度、小数点后位数等。

In [7]:

```
print('1          10          20          30')
print('----+----|----+----|----+----|')
print(f'{3.1415926535897932384626:30.8}')
print(f'{3.1415926535897932384626:<30.8}')
print('1          10          20          30')
print('----+----|----+----|----+----|')
print('Pi = {:<30.8}'.format(3.1415926535897932384626))
print('Pi = {:<30.8}'.format(3.1415926535897932384626))
```

```
1          10          20          30
----+----|----+----|----+----|
                        3.1415927
3.1415927
1          10          20          30
----+----|----+----|----+----|
Pi =                        3.1415927
Pi = 3.1415927
```

- 10. 各种运算符
 - + - 数值的运算、字符串、列表等的拼接
 - * - 数值的运算、字符串、列表等的重复
 - / - 除法
 - // - 整除
 - % - 取余
 - in - 字符串、列表、元组、集合、字典等成员资格的判断
 - += , -= , *= , /= , //= , %= - 扩展的赋值运算符

In [8]:

```
# `+` - 数值的运算、字符串、列表等的拼接
print(1231+999999)           # 整数相加
print('Hello' + ' ' + 'World!') # 字符串拼接
print([1, 3, 7] + [5, 7, 9, 11]) # 列表拼接
print((1, 3, 7) + (5, 7, 9, 11)) # 元组拼接
```

1001230
Hello World!

```
[1, 3, 7, 5, 7, 9, 11]
(1, 3, 7, 5, 7, 9, 11)
```

In [9]:

```
# `*` - 数值的运算、字符串、列表等的重复
print(123321 * 3)           # 整数相乘
print('哈' * 7)             # 字符串重复
print([1, 2, 3] * 5)        # 列表重复
print([[1, 2, 3]] * 5)      # 嵌套列表重复
print((1, True, "Good") * 3) # 元组重复
print(((1, True, "Good"),) * 3) # 嵌套的元组重复
print(('One',) * 3)         # 一个元素的元组重复
```

```
369963
哈哈哈哈哈
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]
[[1, 2, 3], [1, 2, 3], [1, 2, 3], [1, 2, 3], [1, 2, 3]]
(1, True, 'Good', 1, True, 'Good', 1, True, 'Good')
((1, True, 'Good'), (1, True, 'Good'), (1, True, 'Good'))
('One', 'One', 'One')
```

In [10]:

```
# `/` - 除法
# `//` - 整除
# `%` - 取余
# 100 / 7 = 14 ... 2
a = 100 / 7
b = 100 // 7
c = 100 % 7
print(a, b, c, sep = ', ')
```

```
14.285714285714286, 14, 2
```

In [11]:

```
# `in` - 字符串、列表、元组、集合、字典等成员资格的判断
a = 'a' in 'abc'      # 字符串
b = 'lo' in 'hello'   # 字符串
c = 1 in [1, 2, 3]    # 列表
d = 1 in (1, 2, 3)    # 元组
e = 1 in {1, 2, 3}     # 集合
f = 1 in {1: 'one', 2: 'two', 3: 'three'} # 字典键
g = 'one' in {1: 'one', 2: 'two', 3: 'three'}.values() # 字典值
print(a, b, c, d, e, f, g, sep = ', ')
```

```
True, True, True, True, True, True, True
```

In [12]:

```
# `+=`, `-=`, `*=`, `/=`, `//=`, `%=` - 扩展的赋值运算符
x = y = 100
x = x + 10
y += 10
print('100 + 10 ->', x, y)

m = n = 100
m = m // 7
n //= 7
print('100 // 7 ->', m, n)
```

```
100 + 10 -> 110 110
100 // 7 -> 14 14
```

- 11. 内置函数

函数

用途

函数	用途
<code>pow(base, exp)</code>	<code>base ^ exp</code>
<code>len(s)</code>	返回对象的长度（元素个数）。实参可以是序列（如 <code>string</code> 、 <code>bytes</code> 、 <code>tuple</code> 、 <code>list</code> 或 <code>range</code> 等）或集合（如 <code>dictionary</code> 、 <code>set</code> 或 <code>frozen set</code> 等）。
<code>eval(expression)</code>	表达式解析参数 <code>expression</code> 并作为 Python 表达式进行求值。
<code>sum(iterable, start = 0)</code>	从 <code>start</code> 开始自左向右对 <code>iterable</code> 的项求和并返回总计值。 <code>iterable</code> 的项通常为数字，而 <code>start</code> 值则不允许为字符串。
<code>int(x, base=10)</code>	返回一个基于数字或字符串 <code>x</code> 构造的整数对象，或者在未给出参数时返回 0。
<code>float(x)</code>	返回从数字或字符串 <code>x</code> 生成的浮点数。
<code>str(object='')</code>	返回一个 <code>str</code> 版本的 <code>object</code> 。
<code>list([iterable])</code>	可以用多种方式构建列表
<code>tuple([iterable])</code>	可以用多种方式构建元组
<code>set([iterable])</code>	返回一个新的 <code>set</code> 或 <code>frozenset</code> 对象，其元素来自于 <code>iterable</code> 。
<code>dict()</code>	返回一个新的字典，基于可选的位置参数和可能为空的关键字参数集来初始化。
<code>zip(*iterables, strict=False)</code>	在多个迭代器上并行迭代，从每个迭代器返回一个数据项组成元组。
<code>enumerate(iterable, start=0)</code>	返回一个枚举对象。 <code>iterable</code> 必须是一个序列，或 <code>iterator</code> ，或其他支持迭代的对象。

In [13]:

```
a = pow(10, 2)
b = pow(10, 2) % 7
c = pow(10, 2, 7)
print(a, b, c, sep = ',')
```

100, 2, 2

In [14]:

```
a = len('abc')           # 字符串
b = len([1, 2, 3])       # 列表
c = len((1, 2, 3))       # 元组
d = len({1, 2, 3})       # 集合
e = len([(1, 2, 3), ('foo', 'bar')]) # 元组的列表
f = len([(1, 2, 3), ('foo', 'bar')][0]) # 元组的列表的第0个元素
g = len([(1, 2, 3), ('foo', 'bar')][1]) # 元组的列表的第1个元素
print(a, b, c, d, e, f, g, sep=',')
```

3, 3, 3, 3, 2, 3, 2

In [15]:

```
x = eval('100')
y = eval('3.1415926')
z = eval('x * y')
print(x, y, z, sep = ',')
```

100, 3.1415926, 314.15926

In [16]:

```
a = sum((1, 2, 3)) # 元组
b = sum([1, 2, 3]) # 列表
c = sum({1, 2, 3}) # 集合
d = sum({1: 'one', 2: 'two', 3: 'three'}) # 字典键
e = sum({'one': 1, 'two': 2, 'three': 3}.values()) # 字典值
print(a, b, c, d, e, sep = ',')
```

6, 6, 6, 6, 6

In [17]:

```
print(list(range(10)))
print(list({1, 2, 3}))
print(tuple([1, 2, 3]))
print(set([1, 2, 3]))

d = {1: 'one', 2: 'two', 3: 'three'}
print(list(d))
print(list(d.values()))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3]
(1, 2, 3)
{1, 2, 3}
[1, 2, 3]
['one', 'two', 'three']
```

In [18]:

```
# 构建字典
a = dict(one=1, two=2, three=3)
b = {'one': 1, 'two': 2, 'three': 3}
c = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
d = dict([('two', 2), ('one', 1), ('three', 3)])
e = dict({'three': 3, 'one': 1, 'two': 2})
f = dict({'one': 1, 'three': 3}, two=2)
print(a, b, c, d, e, f, sep='\n')
```

```
{'one': 1, 'two': 2, 'three': 3}
{'one': 1, 'two': 2, 'three': 3}
{'one': 1, 'two': 2, 'three': 3}
{'two': 2, 'one': 1, 'three': 3}
{'three': 3, 'one': 1, 'two': 2}
{'one': 1, 'three': 3, 'two': 2}
```

In [19]:

```
a = list(zip([1, 2, 3], ['one', 'two', 'three']))
b = list(enumerate(['one', 'two', 'three']))
c = list(enumerate(['one', 'two', 'three'], 1))
print(a, b, c, sep = '\n')
```

```
[(1, 'one'), (2, 'two'), (3, 'three')]
[(0, 'one'), (1, 'two'), (2, 'three')]
[(1, 'one'), (2, 'two'), (3, 'three')]
```

- 12. 字符串的正向反向索引、切片（左闭右开）的应用

Index from front	0	1	2	3	4	5	6	7	8	9	10	11
Index from back	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
String	a	b	c	d	e	f	g	h	i	j	k	l

In [20]:

```
a = 'abcdefghijkl'[3]
b = 'abcdefghijkl'[-3]
c = 'abcdefghijkl'[3:7]
d = 'abcdefghijkl'[3:-2]
e = 'abcdefghijkl'[-10:-2]
f = 'abcdefghijkl'[3:]
g = 'abcdefghijkl'[:3]
h = 'abcdefghijkl'[:]
i = 'abcdefghijkl'[3:10:2]
j = 'abcdefghijkl'[3:-2:2]
k = 'abcdefghijkl'[::-1]
```

```
l = 'abcdefghijkl'[::-2] # 从开始到结尾, - 代表倒序, 步长2
print(a, b, c, d, e, f, g, h, i, j, k, l, sep = '\n')
```

```
d
j
defg
defghij
cdefghij
defghijkl
abc
abcdefghijkl
dfhj
dfhj
lkjihgfedcba
ljhfdb
```

- 13. 字符串方法 - split() (只要求能读懂程序)

```
str.split(sep=None, maxsplit=- 1)
```

返回一个由字符串内单词组成的列表，使用 sep 作为分隔字符串。如果给出了 maxsplit，则最多进行 maxsplit 次拆分（因此，列表最多会有 maxsplit+1 个元素）。如果 maxsplit 未指定或为 -1，则不限制拆分次数（进行所有可能的拆分）。

In [21]:

```
strEng = 'the quick brown fox jumps over the lazy dog'
strChn = '敏捷的棕色狐狸跳过了懒狗'
a = strEng.split()
b = strEng.split(sep = 'o')
c = strChn.split()
d = strChn.split(sep = '的')
print(a, b, c, d, sep = '\n')
```

```
['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
['the quick br', 'wn f', 'x jumps ', 'ver the lazy d', 'g']
['敏捷的棕色狐狸跳过了懒狗']
['敏捷', '棕色狐狸跳过了懒狗']
```

- 14. 程序控制结构
 - 分支结构（单分支、双分支、多分支）
 - 循环结构的相关语法
 - 应用
 - if ... elif ... else
 - for
 - while
 - break & continue
 - for ... else / while ... else

In [22]:

```
for num in range(-3, 3, 1):
    print('{:2} is a '.format(num), end = '')
    if num > 0:
        print("Positive number")
    elif num == 0:
        print("Zero")
    else:
        print("Negative number")
```

```
-3 is a Negative number
```

```
-2 is a Negative number
-1 is a Negative number
0 is a Zero
1 is a Positive number
2 is a Positive number
```

In [23]:

```
for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print(n, 'equals', x, '*', n // x)
            break
        else:
            # loop fell through without finding a factor
            print(n, 'is a prime number')
```

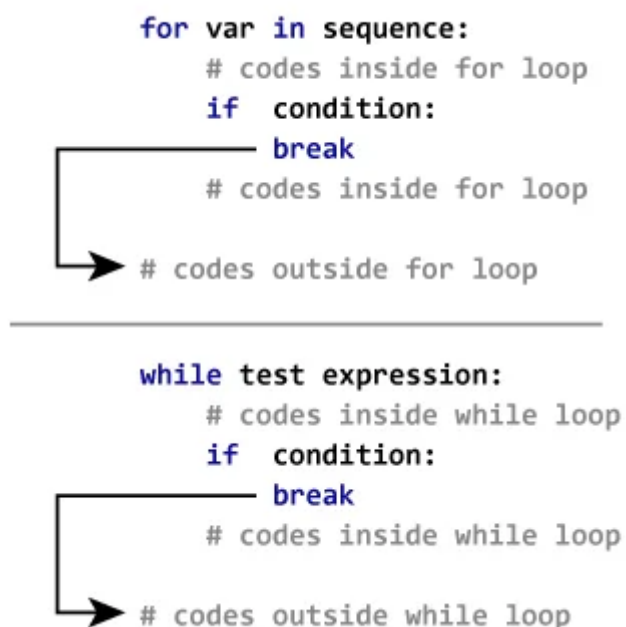
```
2 is a prime number
3 is a prime number
4 equals 2 * 2
5 is a prime number
6 equals 2 * 3
7 is a prime number
8 equals 2 * 4
9 equals 3 * 3
```

In [24]:

```
genres = ['pop', 'rock', 'jazz']
for genre in genres:
    print("I like", genre)
```

```
I like pop
I like rock
I like jazz
```

- break 的工作方式



In [25]:

```
for val in "string":
    if val == "i":
        break
    print(val)
```

```
print("The end")
```

```
s
t
r
The end
```

- continue 的工作方式

```
for var in sequence:
    # codes inside for loop
    if condition:
        continue
    # codes inside for loop
```

```
# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if condition:
        continue
    # codes inside while loop
```

```
# codes outside while loop
```

In [26]:

```
for val in "string":
    if val == "i":
        continue
    print(val)

print("The end")
```

```
s
t
r
n
g
The end
```

In [27]:

```
a = 0
while a < 30:
    print(a)
    if a < 3:
        a += 1
    elif a < 30:
        a += 10
    else:
        break
else:
    print('Wow')
```

```
0
1
2
3
13
```


• 15. 布尔表达式的使用

运算	含意
<	严格小于
<=	小于或等于
>	严格大于
>=	大于或等于
==	等于
!=	不等于
is	对象标识
is not	否定的对象标识

运算	结果
x or y	if x is false, then y, else x
x and y	if x is false, then x, else y
not x	if x is false, then True, else False

• 16. 循环语句中可迭代的结构：range、字符串、列表、元组、集合、字典、文件

- range 类型表示不可变的数字序列，通常用于在 for 循环中循环指定的次数。

range(stop)
range(start, stop[, step])

```
In [28]: print(list(range(5)))  
print(list(range(1, 5)))  
print(list(range(0, 30, 5)))
```

[0, 1, 2, 3, 4]
[1, 2, 3, 4]
[0, 5, 10, 15, 20, 25]

```
In [29]: for n in range(5):  
        print(n, end=' ')
```

0 1 2 3 4

```
In [30]: for c in 'string':  
        print(c, end = ' ')
```

s t r i n g

```
In [31]: for l in ['one', 'two', 'three']:  
        print(l)
```

one
two
three

```
In [32]: for k, v in {1: 'one', 2: 'two', 3: 'three'}.items():
        print(k, v)
```

```
1 one
2 two
3 three
```

- 17. 列表操作的方法
 - `append(x)` - 在列表末尾添加一个元素。
 - `pop([i])` - 删除列表中指定位置的元素，并返回被删除的元素。未指定位置时，`a.pop()` 删除并返回列表的最后一个元素。

```
In [33]: fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
print(fruits)
fruits.append('mongo') # 追加一个元素
print(fruits)
fruits.pop(3) # 删除第五个元素 banana
fruits.pop(3) # 删除第五个元素 kiwi
print(fruits)
fruits.pop() # 删除最后一个元素 mongo
print(fruits)
```

```
['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana', 'mongo']
['orange', 'apple', 'pear', 'apple', 'banana', 'mongo']
['orange', 'apple', 'pear', 'apple', 'banana']
```

- 18. 列表的排序方法 `sort()` 以及内置函数 `sorted()` 的应用场合、语法、排序规则的指定 (lambda函数)、返回值等

```
In [34]: # TODO
```

- 19. 元组的非正规写法
 - 多变量赋值
 - 两变量值交换

```
In [35]: # 多变量赋值
a, b, (c, d), e = 1, 3, (5, 7), 9
print(a, b, c, d, e, sep = ', ')
```

```
# 两变量值交换
x, y = 'hello', 'world'
x, y = y, x
print(x, y)
```

```
1, 3, 5, 7, 9
world hello
```

- 20. 元组的基本要求和操作
 - 利用列表里面嵌套元组完成相应应用描述，并能进行操作
- 21. 集合运算符：&，|，-（只要求能读懂程序）

```
In [36]: girls = {'Adele', 'Lady Gaga', 'G.E.M'}
boys = {'Jay Zhou', 'Ed Sheeran', 'Justin Bieber'}
chinese = {'G.E.M', 'Jay Zhou'}
```

```
singers = girls | boys
chineseBoys = chinese & boys
foreignGirls = girls - chinese
print(singers, chineseBoys, foreignGirls, sep = '\n')
```

```
{'Lady Gaga', 'G.E.M', 'Jay Zhou', 'Justin Bieber', 'Adele', 'Ed Sheeran'}
{'Jay Zhou'}
{'Lady Gaga', 'Adele'}
```

- 22. 集合操作的方法: add()

In [37]:

```
girls = {'Adele', 'Lady Gaga', 'G.E.M'}
girls.add('WanTing')
print(girls)
```

```
{'Lady Gaga', 'WanTing', 'Adele', 'G.E.M'}
```

- 23. 集合的去重重复工作

In [38]:

```
l = list('Hello world, my girl!')
```

顺序无关

```
unique = list(set(l))
print(unique)
```

顺序有关

```
unique = []
seen = set()
for e in l:
    if e in seen:
        continue
    else:
        unique.append(e)
        seen.add(e)
print(unique)
```

```
['y', 'd', 'm', 'g', 'l', ',', 'w', 'H', 'i', 'r', ' ', '!', 'e', 'o']
['H', 'e', 'l', 'o', ' ', 'w', 'r', 'd', ',', ' ', 'm', 'y', 'g', 'i', '!']
```

- 24. 字典添加新的键值对

In [39]:

```
d = {'whale': 5, 'shark': 3}
print(d)
d['python'] = 9 # Add 1
print(d)
d.update({'shrimp': 10000, 'kraken': 1, 'serpant': 2}) # Add many
print(d)
```

```
{'whale': 5, 'shark': 3}
{'whale': 5, 'shark': 3, 'python': 9}
{'whale': 5, 'shark': 3, 'python': 9, 'shrimp': 10000, 'kraken': 1, 'serpant': 2}
```

- 25. 字典中键、值、键值对的获取 keys(), values(), items()
 - 对字典元素的迭代默认情况是对键的迭代
 - 能够用字典描述特定键值对类型的应用并操作

In [40]:

```
d = {'whale': 5, 'shark': 3, 'python': 19}
print(d.keys())
```

```

print(d.values())
print(d.items())
for k in d:
    print(k)
for k, v in d.items():
    print('the number of {:<6} = {:>2}'.format(k, v))

```

```

dict_keys(['whale', 'shark', 'python'])
dict_values([5, 3, 19])
dict_items([('whale', 5), ('shark', 3), ('python', 19)])
whale
shark
python
the number of whale = 5
the number of shark = 3
the number of python = 19

```

- 26. 字典的 get() 方法的作用及使用

get(key[, default])

如果 key 存在于字典中则返回 key 的值，否则返回 default。如果 default 未给出则默认为 None，因而此方法绝不会引发 KeyError。

In [41]:

```

d = {'whale': 5, 'shark': 3, 'python': 19}
print(d['whale'])
# print(d['shrimp'])          # shrimp 不存在, 报错
print(d.get('whale'))
print(d.get('shrimp'))        # shrimp 不存在, 不报错
print(d.get('shrimp', 999))   # shrimp 不存在, 返回默认值

```

```

5
5
None
999

```

- 27. 函数定义及简单参数传递

In [42]:

```
# TODO
```

- 28. 使用pip工具查看当前已安装的Python扩展库的完整命令 `pip list`，安装扩展库命令 `pip install` 库名。

- pip list

```

$ pip list
Package                Version
-----
appdirs                1.4.4
appnope                0.1.2
argon2-cffi            21.3.0
argon2-cffi-bindings  21.2.0
attrs                  21.2.0
...
...

```

- pip install

```
$ pip install pandas
$ pip install wordcloud
$ pip install jieba
```

- 29. 文件打开模式： `r` , `w`
- 30. 读文本文件的方法： `read()` , `readlines()`
 - 对文件对象的迭代默认情况是 `readlines()`
- 31. 写文本文件的方法： `writelines()`
- 32. CSV库, json库读写文件操作的基本语法
- 33. 上下文管理器的应用： `with open() as f:`
- 34. 第三方库random的方法： `randint()`

In [43]:

```
import random
random.seed(5)
x = random.randint(0, 9) # 唯一的闭区间[0, 9]
random.seed(5)          # 同样的seed
y = random.randint(0, 9) # 得到同样的随机数
print(x, y)
```

9 9

In [44]:

```
import random
cards = list('A23456789JQK')
random.shuffle(cards)
print(cards)
random.shuffle(cards)
print(cards)
```

```
['3', '4', 'K', 'Q', 'J', '2', '7', '8', 'A', '9', '6', '5']
['9', 'K', '6', '2', '3', '5', 'J', '4', 'A', '7', 'Q', '8']
```

- 35. 可视化：绘制饼图、直方图、多种折线图、散点图

In [45]:

```
# TODO
```

题型

所有选择题和判断题均出自砺儒云平台的课前、课后测试题，只有个别题目做了适当修改

#	题型	分数
1	选择题	10x2
2	判断题	15x1
3	程序分析题	5x5
4	应用题（程序填空）	10x2
5	思维提升题（问答题）	2x5
6	应用创新题（编程题）	1x10