Python 零基础知识手册

变量

什么是变量

在计算机系统中，变量是指存储在内存中的数据，每创建一个变量就会在系统中开辟一个内存空间供其使用。
我们目前不需要了解创建的变量如何在内存中存储，只要知道 i = 1 这是一个变量赋值语句，其中 i 是变量名，这个变量代表 1，中间的等号是赋值运算符。

变量命名规则

- 大小写英文、数字和_的结合，且不能用数字开头；
- 系统关键词不能做变量名使用； 获取关键字列表： help('keywords')

```
>>> help('keywords')

Here is a list of the Python keywords.  Enter any keyword to get more help.

False               def                 if                  raise
None                del                 import              return
True                elif                in                  try
and                 else                is                  while
as                  except              lambda              with
assert              finally             nonlocal            yield
break               for                 not
class               from                or
continue            global              pass
```

- Python 中的变量名区分大小写；
- 变量名不能包含空格，但可使用下划线来分隔其中的单词；
- 不要使用 python 的内置函数名称做变量名；

内置函数				
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()

bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

变量赋值过程

x = 1 编程中的等号不是等于的意思，是 赋值 的意思，把 1 赋值给 x
x = x + 1 所以这个语句是把 x+1 之后的值再赋值给 x，此时 x 的值为 2

进阶的赋值过程

同时给多个变量赋予同一个内容： a = b = c = 100
同时给多个变量赋予不同的内容： a, b, c = 1, 2, 3

数据类型

数字型：数

比如 1 、9999 、5.28 等等；

字符串： ‘ ’ 或者 “ ”

比如 ‘I love China’ 、‘今天天气不错’ 、 “vndihgihb*@。" 等等；

列表：[, ,]

比如 [1,3,5.5,9.0] 、[“my",“name",“is",“Jack"] 、['qq',37] 等等；

元组：(, ,)

比如 (1.0,'two',3.89,'four') 等等；

字典：{ key1 : value1 , key2 : value2}

比如 {‘name'：‘Jack’，‘age’：15，‘country’：‘China’} 等等；

集合：{ , , }

比如 {1,2,3,4,5} 等等；

布尔型：

True 或者 False 。

作业答案 1：

练习 1：精致女孩的午饭钱

```
自动检测

#作为一个精致的办公室男孩/女孩，每天的午餐一定要营养均衡搭配得当。
#每天你有 30 块钱用作午饭，你要买一瓶 4 块钱的饮料，再花 24 块钱买午餐。让我们尝试用编程算一下还会剩下多少钱。
#要求：
#创建一个 my_money 变量，给它设定为 30
#创建一个 drink_cost 变量，给它设定为 4
#创建一个 lunch_cost 变量，给它设定为 24
#输出结果为：
#午餐后剩余钱数为：
#2

my_money = 30
drink_cost = 4
lunch_cost = 24

print('午餐后剩余钱数为：')
print(my_money - drink_cost - lunch_cost)
```

练习 2：起床猫打印机

```
自动检测

#今天早上起来，你发现自己变成了一只猫！
#但是你依然要精神抖擞地开始新的一天噢！
#要求：
#试着把 name 变量改成喵喵
```

```
#把 wakeuphour  改成 10

name = '喵喵'
wake_up_hour = 10


print("zzz ")

print("   <⌒/ヽ-、___")

print(" /<_/________/ ")
print("‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾")
print("")

print(f"        ∧ ∧         我是{name}")
print(f"       (   •ω•)          我每天{wake_up_hour}点起床")
print("     __|   ⊃/(______")
print(" /     └-(________/   ")
print("‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾")
```

数字型

数字型可分为整数（int）和浮点数（float）。

算术运算符

算术运算符主要用于算数计算。
常见的运算符主要有以下几种：

运算符	描述
+	加法运算符
-	减法运算符
*	乘法运算符
/	除法运算符
**	指数运算符

%	取余运算符，计算余数
//	除法取整运算符，计算商并除其小数部分

比较运算符

比较值的大小。

常见的比较运算符主要有以下几种：

运算符	描述
>	判断第一个运算对象是否大于第二个运算对象
<	判断第一个运算对象是否小于第二个运算对象
>=	判断第一个运算对象是否大于或等于第二个运算对象
<=	判断第一个运算对象是否小于或等于第二个运算对象
==	判断两个运算对象是否相同
!=	判断两个运算对象是否不相同

字符串

字符串是由字母、数字和特殊字符来组成的序列。字符串从左到右索引默认 0 开始的，最大范围是字符串长度少 1，从右到左索引默认-1 开始的，最大范围是字符串开头。

如何创建字符串？

使用单引号、双引号或者三引号

如例：

```
自动检测

name='hanmeimei'
number="12"
paragraph='''Hello,makerbean!
Hello,world! '''
```

如何获取字符串的长度？

len()函数：返回字符串中的字符长度或者字符数。
如例：

```
自动检测

s1='hello world'
s2='makerbean'
print(len(s1))
print(len(s2))
```

如何获取字符串中的字符？

以 name='hanmeimei' 为例：
•获取单个字符：

```
自动检测

print(name[0])
```

字符串内置方法

方法	描述
.lower()	把字符串中的大写字母转换成小写字母
.upper()	把字符串中的小写字母转换成大写字母
.find(“x”)	查找 x 字符的第一个下标
.replace(“old”,” new”)	用其他字符串替换字符串
.count(“x”)	返回 x 字符在字符串中出现的次数
.isalpha()	如果字符串只包含字母则返回 true
.isdigit()	如果字符串只包含数字则返回 true

字符串的连接

用“+”将两个字符串连接在一起。

```
自动检测

s1='hello'
```

```
s2='world'
print(s1+s2)#形成了一个字符串
print(s1,s2)
```

读取用户的输入数据

用 input()函数获取，注意得到的数据是字符串类型的。

作业答:2:

喵餐厅的三个套餐

```
自动检测

#今天中午你发现了一家新开业的餐厅——喵餐厅！
#喵餐厅里面看起来有很多好吃的东西呢！
#作为一个会编程的精致男孩/女孩，你决定用编程来算算哪个套餐最划算。
#当然我们衡量划不划算的标准非常简单粗暴：套餐内每克食物的价格越低越划算！
#套餐 1：咖喱鸡肉饭（800g）+可乐（330g） = 28 元
#套餐 2：回锅肉饭（650g）+奶茶（500g） = 30 元
#套餐 3：龙利鱼饭（400g）+小菜（150g）+汤（300g） = 26 元
#输出要求：
#  每个套餐的食物总克数/价格的值
#  按照套餐 1、套餐 2、套餐 3 的顺序分三行输出

#答案
package1 = (800 + 330) / 28
package2 = (650 + 500) / 30
package3 = (400 + 150 + 300) / 26
print(package1)
print(package2)
print(package3)
```

平行宇宙的发型新闻

```
自动检测

#今天，在我们的世界中有这样一条新闻：
```

```
#《纽约邮报》2018 年曾刊发文章说，美国总统特朗普的发型是移植手术加强力发胶固定的结果。该报道援引《烈焰与怒火：特朗普白宫揭秘》的内容称，特朗普的女儿伊万卡喜欢在朋友面前嘲笑父亲的发型，她经常向朋友介绍做这个发型的技巧：植发后，以侧面的和前面的头发包围完全光秃的头顶。
#而平行世界的喵宇宙中，美国总统并不是特朗普而是喵喵！
#你作为喵喵的新闻发言人，需要利用字符串的 replace()函数，将这一段新闻中的特朗普都替换成喵喵并打印出来。
#输出要求：
#  填写 replace()里的参数
#  输出新闻

news = '《纽约邮报》2018 年曾刊发文章说，美国总统特朗普的发型是移植手术加强力发胶固定的结果。该报道援引《烈焰与怒火：特朗普白宫揭秘》的内容称，特朗普的女儿伊万卡喜欢在朋友面前嘲笑父亲的发型，她经常向朋友介绍做这个发型的技巧：植发后，以侧面的和前面的头发包围完全光秃的头顶。'



print(news.replace("特朗普", "喵喵"))
```

布尔表达式

布尔值

• 用于表示判断中的是与否 ，一般用于条件测试中 ；
• 取值只有 True 、False ；

逻辑运算

用于检测两个或两个以上的条件是否满足 ； 逻辑运算只存在与布尔类型中 。
• and ，逻辑“与” 当运算符两边的两个运算对象都为 True 时 ，结果为 True ；
• or ，逻辑“或” 当运算符两边的两个运算对象其中有一个运算对象为 True 时，结果即为 True；
• not ，逻辑“非” 用于反转运算对象的状态。

布尔表达式

实例如下：

```
自动检测

>>>3 and 5
5

>>>3 or 5
3

>>>0 or 5
5

>>>3 and not 5
False
```

表达式的应用——条件测试

- 检查当前变量是否与一个特定值相等/不相等 ；
- 比较数字的大小 ；
- 检查特定值是否在某序列里 。

表达式的应用——多条件检查

- 使用 and 检查多个条件 ：

```
自动检测

age_lilei = 17
age_hanmeimei = 18
age_lilei >= 18 and age_hanmeimei >=18
```

False

- 使用 or 检查多个条件 ：

```
自动检测

age_lilei >= 18 or age_hanmeimei >= 15
```

True

```
自动检测

age_lilei >=20 or age_hanmeimei >= 20
```

False

Python 代码缩进问题

- 用 四个空格 或者一个 Tab 来表示缩进都可以，但是不要混用 ；
- 相同缩进位置的代码表示它们是同一个代码块 ；

if 条件判断

条件判断——if/else

如果 明天下雨
    宅在家打游戏
    否则如果 球场开门
    出去打球
否则
去图书馆自习
明天下雨——>宅在家打游戏
明天不下雨 且 球场开门——>出去打球
明天不下雨 且 球场不开门 ——>去图书馆自习
实例如下（判断用户输入的内容）：

```
自动检测

user_gender = input("请输入您的性别（F/M）： ")
if user_gender == 'F':
print("你是萌妹子")
elif user_gender == 'M': # elif 是 else if 的缩写
print("你是糙汉子")
else：#如果没有 else 语句且前面的条件都不符合则什么都不输出
print("输入不正确，请输入 F 或 M")
```

赋值与判断相等

- 单等号 = 是赋值

比如 ： a = 3 ， 表示把 3 赋值给 a

- 双等号 == 是判断相等

比如 ： if a == 3: ， 表示如果 a 的值等于 3

作业答案 3：

名字越长才会越强

```
自动检测

# 今天下班回家后，你打开游戏遇到了一个名字很长的对手，你觉得不能在名字的气势上就输给对方！
# 于是你决定给自己起一个更长的名字把对手比下去！
```

```
# 会编程的你当然不能自己一个个字去数，于是你决定写一个程序来帮自己判断新名字是否比对手更长。
# #输出要求：
# 给 my_name 变量赋值一个更长的字符串名字
# ?的地方改成大于号或者小于号，想一想应该是大于还是小于？
# 输出内容为：**我的名字更长吗？ True**


# 答案
opponent_name = '妈妈说名字要够长才有气势'
my_name = '111111111111111111111111'
print('我的名字更长吗？', len(my_name) > len(opponent_name))
```

懒人日程安排器

自动检测

```
# 会了编程以后，我们可以少操心很多事情，比如日程安排也可以交给程序来自动安排！
# 今天我们要做一个简单的懒人日程安排器，根据天气情况告诉我们今天该做什么事情。
# - 如果是下雨天就在家里看书
# - 如果是晴天就出去散步
# - 如果是阴天就去逛商场
# 输出要求：
# - 完成 if...elif...else 的编写
# - 使得程序能按照描述进行输出

# 答案
weather = '晴'

if weather == '雨':
    print('在家看书')
elif weather == '阴':
    print('去逛商场')
else:
```

```
    print('出去散步')
```

列 表

列表结构

- 利用中括号表示列表
- 列表内的元素用逗号隔开
- 注意是英文输入法下的逗号，如例：

```
自动检测

student1 = ['lilei',18,'class01',201901]
student2 = ['hanmeimei',19,'class02',201902]
```

列表具有可变性：可以修改列表中的内容

获取列表中的元素

编程语言中通常第一个位置的编号是 0，以此类推。
以下列表为例：

```
自动检测

grade = [98,99,95,80]
```

获取列表的某个元素，通常用中括号将元素的位置括起来。

```
自动检测

print(grade[0])
print(grade[0]+grade[3])
```

运行后显示

```
自动检测

98
178
```

列表常用方法

获取列表长度 len(列表)

```
自动检测

student_list = ['李雷','韩梅梅','马冬梅']
print(len(student_list))
```

获取列表中的元素 找到需要修改的元素编号，列表名[编号]=新值

```
自动检测
```

```
student_list = ['李雷','韩梅梅','马冬梅']
student_list[0] = 'lilei'
```

向列表添加元素 列表名.append(要添加的元素)

```
自动检测

inventory = ['钥匙','毒药']
inventory.append('解药')
```

删除列表元素 del+列表元素删除

```
自动检测

student_list = ['李雷','韩梅梅','马冬梅']
del student_list[0]
```

两个列表相加 列表 1+列表 2

```
自动检测

numbers1 = [0,1,2,3,4]
numbers2 = [5,6,7,8,9]
print(numbers1+numbers2)
```

判断某个元素是否存在于列表中 in

```
自动检测

inventory = ['钥匙','毒药','解药']
if '解药' in inventory:
print('yes')
else：
print('no')
```

获取列表中某个元素的重复次数 用列表.count(元素)来获取

```
自动检测

numbers1 = [0,1,1,2,3,4,1]
print(numbers1.count(1))
```

获取列表中某个元素第一次出现的位置 用列表.index(元素)来获取

```
自动检测

numbers1 = [0,1,1,2,3,4,1] print(numbers1.index(1))
```

字 典

字典的结构

{key:value,key:value}
{键:键值}

• 用花括号表示字典
• 字典内每一项都有两个元素组成：key 和 value，key 和 value 一一对应，同一个键只能有一个对应的值，
• 各个项用逗号隔开
• 键的类型是不可变的。如例：

```
自动检测

phone_numbers = {'李雷':'123456','韩梅梅':'456','马冬梅':'45678'}
print(phone_numbers)
```

访问字典中的数据

变量[key] ， 如例：

```
自动检测

grade = {'李雷':'98','韩梅梅':'99'}
print (grade['李雷']) #访问字典里的数据
```

更新字典的元素

变量[key] = 新值

```
自动检测

grade = {'李雷':'98','韩梅梅':'99'}
grade['韩梅梅'] = 100 #更新字典的元素
print(grade)
```

字典的删除操作

```
自动检测

grade = {'李雷':'98','韩梅梅':'99','马冬梅':'95'} del grade['李雷']
#删除字典里的某项  grade.clear() del grade
#删除字典本身  del grade
```

作业答案 4：

群聊关键人名警报机

```
自动检测

# 我们每天都在用聊天工具，群聊有时候太多可能都看不过来。既然我们会编程，不如让机器人帮我们实时监测吧！
```

```
# 监测人员名单：小红，张三，李四，王五，赵六，小明
# 输出要求：
# - 将 user_name 赋值为**小明**
# - 将 names_list 赋值为列表，里面储存描述中的人员名单
# - 判断 user_name 是否在监测列表**names_list**内
# - 如果在列表内则输出：**发现监测目标**
# - 否则输出：**该用户不是监测目标**

# 答案
user_name = ''
names_list = ['小红', '张三', '李四', '王五', '赵六', '小明']

if user_name in names_list:
    print('发现监测目标')
else:
    print('该用户不是监测目标')
```

平均成绩计算器

自动检测

```
# 你的小学体育老师在听说你会编程以后，马上找到你，希望你帮他做一个程序自动计算学生语文成绩的平均分。会了编程的你欣然接受。
# 输出要求：
# - 给 grade 字典添加一个新的元素，索引（key）是人名：小明，数据（value）是 98
# - 统计字典中全部同学的成绩平均分

# 答案
grade = {'小红': 90, '李雷': 85, '韩梅梅': 95, '马冬梅': 93}
grade['小明'] = 98

average_score = (grade['小红'] + grade['李雷'] + grade['韩梅梅'] + grade['马冬梅'] + grade['小明']) / 5
print(average_score)
```