# Python Cheat Sheet-1. 善用print()，基础运算，变量及赋值

?> PCS-1（    ）：1.1 不知道运行结果，就很难写代码——善用print()，第一个Python代码；1.2 增加注释的必要性；1.3 基本的数据类型（Basic Data Types）及运算（Operations）；1.4 变量及赋值 (Variables and assignment)

知识点

描述

代码段

运算结果

备注

**1.1** 不知道运行结果，就很难写代码——善用print()；第一个Python代码

print()是Python语言中使用最为频繁的语句，在代码编写、调试过程中，要不断的用该语句来查看数据的值、数据的变化、数据的结构、变量所代表的内容、监测程序进程、和显示结果等。通过print()实时查看数据反馈，才可知道代码编写的目的是否达到，并做出反馈。善用print()是用Python写代码的基础。

print('Hello World!')代码（打印显示一行字符串）基本成为所有类型编程语言第一行代码的标配，标志着正式开启代码学习的篇章。

print('Hello World!')

Hello World!

实际应用print()时，通常是打印变量来查看变量值，在代码调试时可以直接打印变量，例如print(v\_sum)，下述代码则增加了解释的字符串配合打印变量，使用的是字符串格式化方法中的f"{}"方式。

v\_1=10  
v\_2=7  
v\_sum=v\_1+v\_2  
print(f"The result of the calculation is {v\_sum}")

The result of the calculation is 17.

用print()查看当前Python版本。这里调入了Python的一个[标准库](https://docs.python.org/3/library/index.html)①[platform](https://docs.python.org/3/library/platform.html?highlight=platform#module-platform)②的python\_version方法查看Python版本。调入库的方法，使用import， 如果仅调入库中的一个方法，则可以使用from "library name" import "method"。

from platform import python\_version  
print(python\_version())

3.8.13

print("\_"\*50)，这里对字符"\_"乘以了一个数字，则复制该字符多少个；对于字符串可以使用双引号，也可以使用单引号。但是希望内部字符包括单引号时，则外部使用双引号，而内部使用单引号。如果内容包括双引号时，则需要借助转义字符（escape character）\实现转义，即将Python的特殊字符，例如表征字符串的双引号转换为普通字符串使用。当然，也可以配合使用三引号。如果语句位于同一行，直接可以用;号分割。但是通常不会这么做，因为这使得代码的可读性变弱；右斜杠（backslash，\）可以将长文本切为多段输入，输出字符串不断行。

print("Hello Python!")  
print("\_"\*50)  
print("编程让设计更具'创造力！'");print("Everybody should learn how to code a computer, because it teaches you how to think, and allows designers more creative!")  
print("成为工具的\"建构者！\"")  
print("""You must "type" each of these excercises in, mannually. \  
If you copy and paste, you might as well as not even do them.""")

Hello Python!  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
编程让设计更具'创造力！'  
Everybody should learn how to code a computer, because it teaches you how to think, and allows designers more creative!  
成为工具的"建构者！"  
You must "type" each of these excercises in, mannually. If you copy and paste, you might as well as not even do them.

**1.2** 增加注释的必要性

注释包括单行注释，使用井号（hash，#）开头；多行注释，使用''' comments '''，或者""" comments """。注释并不会被执行，解释器将忽略注释的所有内容。注释的目的：

1. 为作者的注解，方便日后查看已经写过的代码含义，避免重新解读（尤其对于复杂或不易理解的逻辑和算法）；
2. 方便交流，他人阅读该代码时，可以快速的知道代码书写的目的或逻辑；
3. 传递代码书写作者、日期、版权等辅助信息；
4. 书写函数时，以注释的方式说明函数的功用，输入参数和返回变量的数据类型及说明等。

注：用于函数说明时，如果是使用Spyder交互式解释器编写代码，函数名行后回车，会提示是否书写函数说明，并自动配置下述格式，作者仅需要输入必要信息。

# 1-作者备忘注释，及说明方便交流  
data\_path='./data' # 配置数据存储位置  
  
# 2-辅助信息  
  
"""  
Created on Tue Feb 15 09:58:38 2022  
  
@author: Richie Bao-caDesign设计(cadesign.cn)  
"""  
  
# 3- 用于函数说明  
def cfg\_load\_yaml(ymlf\_fp):  
 '''  
 读取 yaml 格式的配置文件  
  
 Parameters  
 ----------  
 ymlf\_fp : string  
 配置文件路径  
  
 Returns  
 -------  
 cfg : yaml-dict  
 读取到python中的配置信息  
 '''  
 import yaml  
 with open (ymlf\_fp,'r') as ymlfile:  
 cfg=yaml.safe\_load(ymlfile)   
 return cfg

**1.3** 基本的数据类型（Basic Data Types）及运算（Operations）

代码处理的对象就是数据，基本的数据类型包括整数（Integer,int），实数（浮点型）（Real numbers, float），复数（Complex numbers，complex），字符（String, str）和布尔（Boolean，bool）。各种数据类型，都可以通过Python内置函数（方法）type查看数据类型。

注：内置函数为可以直接调用的函数，直接使用而无需导入库（模块）。

print(type(7))  
print(type(3.1415926))  
print(type(3+6j))  
print(type('Small is Beautiful'))  
print(type(True),type(False))

<class 'int'>  
<class 'float'>  
<class 'complex'>  
<class 'str'>  
<class 'bool'> <class 'bool'>

* 变换数据类型

int(value,base)，其中base基数默认为10。float(value)只有一个输入参数。可已用内置函数转二进制、十进制和十六进制，其计算结果类型表述中0b代表二进制，0o代表八进制，ox代表十六进制。

print(int(3.1415926))  
print(int(2.7182818)) # 直接使用int()会自动向下取整  
print(int("255",10)) # 字符串转整数。如果字符串内容为浮点数，则会提示错误  
  
print(bin(12)) # 转二进制（binary）  
print(oct(12)) # 转十进制（octal）  
print(hex(12)) # 转十六进制（hexadecimal）  
  
print("\_"\*50)  
print(float(64))  
print(float("1.618034"))  
  
print("\_"\*50)  
print(complex(10))  
print(complex("10+3j"))  
  
print("\_"\*50)  
print(bool(0))  
print(bool(1))  
print(bool())  
print(bool(""))  
print(bool("values"))  
  
print("\_"\*50)  
print(str(3.1415926),":",type(str(3.1415926)))

3  
2  
255  
0b1100  
0o14  
0xc  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
64.0  
1.618034  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
(10+0j)  
(10+3j)  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
False  
True  
False  
False  
True  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
3.1415926 : <class 'str'>

* 运算类型（Types of Operators）

6+7=13中，数值（numerical values）6和7为操作数（operands）；+为运算符/操作符（operators）。

1. 算数运算符

| 运算（Syntax） | 说明 (Description) |
| --- | --- |
| a+b | a加b (Addition) |
| a-b | a减b (Subtraction) |
| a\*b | a乘以b (Multiplication) |
| a/b | a除以b (Division) |
| a//b | a除以b后向下取整 (Floor Divisiont) |
| a\*\*b | a的b次方 (Exponential/Power) |
| a%b | 模运算（Modulus）。取模运算是计算两个数相除之后的余数 |

print(15//7)  
print(15%7)

2  
1

1. 比较运算符（Comparison/Relational Operators）

比较运算结果为布尔值（True 或False）。

| 运算（Syntax） | 说明 (Description) |
| --- | --- |
| a>b、a>=b | 如果a大于（或大于等于）b，则结果为True (Greater than, Greater than or equal to) |
| a<b、a<=b | 如果a小于等于（或小于）b，则结果为True (Lesser than, Lesser than or equal to) |
| a==b | 如果a等于b，则结果为True (Equal to) |
| a!=b | 如果a不等于b，则结果为True (Not equals to) |

print(6!=7)  
print(6==7)  
  
print("\_"\*50)  
print("six"!="seven")  
print("six"=="six")  
  
print("\_"\*50)  
print(2.718==2.718000)

True  
False  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
True  
True  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
True

1. 赋值运算符（Assignment Operators）

赋值运算符相当于将等号右边的值按运算符计算到等号左边值，此时a为变量，而不是具体的值，计算后的值再赋值给变量a。

| 运算（Syntax） | 等价于（Syntax Equivalence） |
| --- | --- |
| a+=b | a=a+b |
| a-=b | a=a-b |
| a\*=b | a=a\*b |
| a/=b | a=a/b) |
| a//=b | a=a//b |
| a\*\*=b | a=a\*\*b |
| a%=b | a=a%b |

i=0  
i+=1  
print(i)  
i+=1  
print(i)

1  
2

1. 逻辑运算符 （Logical Operators）

| 运算（Syntax） | 说明 (Description) |
| --- | --- |
| a and b | 都为True时，返回True |
| a or b | 至少一个为True时，返回True |
| not a | 为True时返回False，为False时返回True |

print(True and True)  
print(True and False)  
print(True or True)  
print(True or False)  
print(False and False)  
print(not True)  
print(not False)

True  
False  
True  
True  
False  
False  
True

1. 按位运算符（Bitwise Operators）

按位运算符通常用于嵌入式系统，多个输入输出端口（高低电平）表示的命令操作中，在数据分析领域使用暂不常见。但&和|可以替代and和or逻辑运算符使用。

| 运算（Syntax） | 说明 (Description) |
| --- | --- |
| a & b | 如果a和b均为True，则结果为True。对于整数（二进制），执行按位与操作。(Bitwise AND) |
| a | b | 如果a和b任意一个为True，返回True。对于整数（二进制），执行按位或操作。(Bitwise OR) |
| a^b | 为True时返回False，为False时返回True。对于布尔值，如果a或b为True（但不都为True），则结果为True。对于整数（二进制），执行按位异或操作。 (Bitwise XOR) |
| ~a | 对于整数（二进制），执行按位取反操作。(Bitwiese NOT) |
| a<<b | 对于整数（二进制），对a执行按位左移b个位操作。(Bitwise left shift) |
| a>>b | 对于整数（二进制），对a执行按位右移b个位操作。(Bitwise right shift) |

print(True & True)  
print(True & False)  
print(True | True)  
print(True | False)  
print(False | False)  
  
print("\_"\*50)  
print(bin(7))  
print(bin(0b0111))  
print(0b0111) # 会自动转换为十进制  
  
print("\_"\*50)  
print(bin(~0b0111))  
  
print("\_"\*50)  
print(bin(0b0111<<1))  
print(bin(0b0111>>1))

True  
False  
True  
True  
False  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
0b111  
0b111  
7  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
-0b1000  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
0b1110  
0b11

1. 成员运算幅 （Membership Operators）

用于判断一个对象是否在Python序列中（例如，string/字符串、list/列表、tuple/元组和array/数组等）。

| 运算（Syntax） | 说明 (Description) |
| --- | --- |
| a in b | 如果a在序列b中，则为True |
| a not in b | 如果a不在序列b中，则为True |

lst=[1,3,6,7,9]  
string="python supports two membership operators, in and not in."  
  
print(2 in lst)  
print(3 in lst)  
print("in" in string)  
print("is not" not in string)

False  
True  
True  
True

1. 同一运算符 （Identity Operators）

用于判断两个对象（例如变量）是否使用同一位置索引内存。可用内置函数id()查看对象唯一标识，即获取对象的内存地址。

| 运算（Syntax） | 说明 (Description) |
| --- | --- |
| a is b | 如果变量a和b指向同一个Python对象，则结果为True |
| a is not b | 如果变量a和b指向不同的Python对象，则结果为True |

a=7  
b=a  
c=7  
d=9  
  
print(b is a)  
print(c is a)  
print(d is a)  
print(id(a),id(b),id(c),id(d))

True  
True  
False  
140730917330880 140730917330880 140730917330880 140730917330944

* 运算符优先级 （Precedence and Associativity Rule of Operators）

包括多个运算符时，优先顺序如下表。L2R表示Left to right（从左到右）；R2L表示Right to left（从右到左）

| 运算符（Operator） | 说明（Description） | 结合性（Associativity） |
| --- | --- | --- |
| () \*\* | 圆括号（Parentheses）幂（Exponential/Power） | L2R R2L |
| +x,-x,~x \*,/,//,% | 一元加（Unary Addition），一元减（Unary Subtraction）, 按位取反（Bitwise NOT）乘（Multiplication），除(Division), 向下取整除（Floor Division）,取模运算（Modulus） | L2RL2R |
| +,- | 算数加（Arithmetic addition）,算数减（Arithmetic subtraction） | L2R |
| <<,>> | 按位左移（Bitwise shift left）, 按位右移（Bitwise shift right） | L2R |
| & | 按位与（Bitwise AND） | L2R |
| ^ | 按位或（Bitwise OR） | L2R |
| |==,!=,>,>=,<,<= | 按位异或（Bitwise XOR）比较运算符（Relational operators） | L2RL2R |
| =,+=,-=,\*=,/=,//=,\*\*=,%= in, not in, is, is not | 赋值运算符（Assignment and Augmented assignment operators））成员，同一运算符（Membership, Identity operators） | R2LL2R |
| Not And | 逻辑非（Logical NOT） 逻辑与（Logical AND） | L2RL2R |
| Or | 逻辑或（Logical OR） | L2R |

x=7  
print((x\*\*2-2\*x-3)/2)

16.0

**1.4** 变量及赋值 (Variables and assignment)

代码读起来应该像流畅的英语散文，而不是加密的密码。好的易读的变量名的定义正是让代码变的流畅的基础。变量名不能以数字和特殊字符为开头，也不可以内置的函数名定义，也不存在空格，如果由几个单词或数字组成变量名，通常由下划线连接，或者每一新单词首字母大写（通常尽量保持一种风格）。变量名定义不符合规范时，解释器会提示错误。

func=2\*y+1 # 当程序逐行从上至下运行时，注意变量定义的顺序  
y=5  
print(func)

11

如果想把变量值作为字符串的一部分打印出来，可以使用字符串格式化方法，例如%形式，或者'{}'.format(variable)方式，及f"{}"方法。

x=5.0  
monadic\_equation=2\*x+1  
print("monadic\_equation=",monadic\_equation)  
print("monadic\_equation=%.2f"%monadic\_equation) # %字符串格式化方法  
print("monadic\_equation={:.2f}".format(monadic\_equation)) # format()字符串格式化方法  
print(f"monadic\_equation={monadic\_equation:.2f}") # f"{}" 字符串格式化方法

monadic\_equation= 11.0  
monadic\_equation=11.00  
monadic\_equation=11.00  
monadic\_equation=11.00

city\_name="Xi'an"  
coordinate\_longitude=108.942292  
coordiante\_latitude=34.261013  
print("The longitude of the Xi'an coordinate is {lon:.2f}, and the latitude is {lat}.".format(lon=coordinate\_longitude,lat=coordiante\_latitude))

The longitude of the Xi'an coordinate is 108.94, and the latitude is 34.261013.

x,y,b=2,5,7 # 序列解包（unpacking）。尝试，x,y,\*z=0,1,2,3,4,5,6; x,y,\*z=0,1; (x,y),(a,b)=(0,1),(2,3)  
func\_2=2\*x+3\*y+b  
print("func\_2={}".format(x,y,b,func\_2))

func\_2=2

当代码变量不断的增多时，可以在尽量保持风格统一条件下，综合下划线和首字母大写，及专业术语缩写等方式表达。

landuseName='General\_Industrial'  
landuseID=3  
GIndustrial\_area=5700  
GIndustrial\_greenArea=3214   
GIndustrial\_GSR=GIndustrial\_greenArea/GIndustrial\_area\*100 # green space ratio（GSR）  
print("GIndustrial\_GSR={:.3f}%".format(GIndustrial\_GSR))

GIndustrial\_GSR=56.386%

注释（Notes）：

① 标准库，The Python Standard Library，Python 标准库非常庞大，提供的组件涉及范围十分广泛，包含了多个内置模块 (以 C 编写)，可以用来实现系统级功能，例如文件 I/O；也有大量Python编写的模块，提供了日常编程中许多问题的标准解决方案；其中有些模块经过专门设计，通过将特定平台功能抽象化为平台中立的 API 来鼓励和加强 Python 程序的可移植性。Windows 版本的 Python 安装程序通常包含整个标准库，往往还包含许多额外组件。对于类 Unix 操作系统，Python 通常会分成一系列的软件包，因此可能需要使用操作系统所提供的包管理工具来获取部分或全部可选组件（<https://docs.python.org/3/library/index.html>）。

② platform， Access to underlying platform’s identifying data，访问底层平台的识别数据 （<https://docs.python.org/3/library/platform.html?highlight=platform#module-platform>）。

PC1-ipynb download