# Python Basic

此教程针对Python编程语言感兴趣的人，从零基础到入门。

Learn Python within 24 hours and learn it well

## Author

@ZL, 20220130

## Changelog

- v0.01, initial build. 20220130

## Content

The following is the skeleton of this course.

1. 配置开发环境 development environment configuration
2. 变 Variable
3. 函 Function
4. 达 Statement and expression
5. 控 Control flow
6. 流 Loop
7. 类 Class
8. 结 Data Structure

# 0. 配置开发环境 development environment configuration

## Windows

1. 官网下载最新Python3编译器。 https://www.python.org/downloads/
2. 设置环境变量。 https://blog.csdn.net/CatStarXcode/article/details/79715530
3. 设置pypi镜像地址。 https://www.jianshu.com/p/e2dd167d2892

Macos: 同Windows，或用brew install Python3

## 如何确认配置成功?

命令行

```
python --version

pip --version

where python
```

# 1. 变 Variable

## What

变量就像一个盒子，或者容器。用来装东西。

哪类? 叫什么? 有多少?

## Why

避免重复。DRY

## How

```
variable_name:Type = value
```

## variable_name

约定俗成

- 全部小写
- 如果由多个单词组成，用下划线连起来
- 不可使用数字开头/Python语言保留关键字/特殊符号等

类型：Python有4类string, int, float, boolean

值：类型和值要匹配

## 类型之间转换

| - | str | int | float | bool |
|-------|-----|-----|-------|------|
| str | - | O | O | O |
| int | O | - | O | O |
| float | O | O | - | O |
| bool | O | O | O | - |

## 输出格式化

口诀："填对宽，分精类"

1. 填充: 不足的空位用指定字符填充
2. 对齐: 左中右
3. 宽度: 整体的宽度
4. 分号: 千分号
5. 精度: 精确到小数点后多少位
6. 类型: int, float, decimal, binary, oct, hex

# 值

科学计数法，或下划线隔开特大数值

## 打印格式化的方法

```
n:int = 42

print('number n: %d' % n) # C style

print('number n: ' + str(n)) # concatenation

print(f'number n: {n}') # f-string
```

In [7]:
```
x = 42
```

In [8]:
```
type(x)
```

Out[8]: int

In [9]:
```
str(x)
```

Out[9]: '42'

In [10]:
```
int('42')
```

Out[10]: 42

In [6]:
```
x + y
x - y
x * y
x / y
```

Out[6]: 111

In [2]:
```
42 - 69
```

Out[2]: -27

In [3]:
```
42 * 69
```

Out[3]: 2898

In [4]:
```
42 / 69
```

Out[4]: 0.6086956521739131

In [ ]:

In [1]:
```python
name:str = 'ZL' # 声明一个叫 name的变量，存储的东西是字符串，值是'ZL'

print(name) # 打印出变量name
```

ZL

In [11]:
```python
print("hello world")
```

hello world

In [12]:
```python
a:int = 69
print(a)
```

69

In [13]:
```python
b:float = 3.14159
print(b)
```

3.14159

In [4]:
```python
c:bool = True
print(c)
```

True

In [5]:
```python
x:str = '42'
type(x)
```

Out[5]: str

In [6]:
```python
int(x)
```

Out[6]: 42

In [7]:
```python
float(x)
```

Out[7]: 42.0

In [8]:
```python
bool(x)
```

Out[8]: True

In [16]:
```python
y:int = 100_000_000
z:int = 3e8

print(y)
print('{0:<20,.2f}'.format(y))
```

100000000
100,000,000.00

In [10]:
```python
print('{0:*<20}'.format(z))
```

300000000.0*********

In [11]:
```python
print('{0:<b}'.format(y))
```

10111110101110000100000000

In [12]:
```python
print('{0:<o}'.format(y))
```

575360400

In [13]:
```python
print('{0:<x}'.format(y))
```

5f5e100

In [14]:
```python
print('%d' % y)
```

100000000

In [15]:
```python
print(f'{y!r}')
```

100000000

# 2. 函 Function

## What

输入 -> 函数 -> 输出。跟数学里的函数概念一样

Note: EFMA

## Why

将可重复使用的代码块整合到一个函数里，不用每次都写。DRY

## How

函数组成：函数名字，参数，返回值

```
def function_name(*args:Any, **kwargs:Any) -> Any:
    ...
    return
```

## 分类

- 普通函数: 掌握✋
- 匿名函数: 掌握✋
- 立即函数: 了解

## 函数名字

跟变量variable约定类似

## 参数

- 位置参数: args

- 关键字参数: kwargs

## 返回值

- 可以返回1个或多个值: 掌握✋
- 也可以不返回任何值: 掌握✋
- 返回对象: 了解
- 返回函数: 了解

In [20]:
```python
def summation_01(a:int, b:int) -> int:
#     rv = a + b
    return a + b
```

In [21]:
```python
summation_01(4, 5)
```

Out[21]: 9

In [23]:
```python
def summation_02(a:int, b:int, c:int) -> int:
    return a + b + c
```

In [24]:
```python
summation_02(4, 2, 3)
```

Out[24]: 9

In [25]:
```python
f = lambda x, y: x + y
```

In [26]:
```python
f(1, 2)
```

Out[26]: 3

In [22]:
```python
(lambda x, y: x * y)(4, 5)
```

Out[22]: 20

In [23]:
```python
hasattr(f, '__call__')
```

Out[23]: True

In [27]:
```python
def add(a:float, b:int=10)->float:
    return a + b
```

In [29]:
```python
add(3.14, 100)
```

Out[29]: 103.14

In [31]:
```python
def sub(a:float=2.718, b:int=3.14)->float:
    return a - b
```

In [33]:
```python
round(sub(),2)
```

Out[33]: -0.42

In [28]:
```python
def general_sum(*args, **kwargs)->float:
    return sum(args) + sum(kwargs.values())
```

In [29]:
```python
general_sum(1, 2, 3, x=1, y=2)
```

Out[29]: 9

In [39]:
```python
def return_nothing():
    print('this function returns nothing')
    return 1
```

In [42]:
```python
x = return_nothing()
```

this function returns nothing

In [43]:
```python
print(x)
```

1

In [31]:
```python
def return_multiple_value():
    return (1, 2, 3)
```

In [32]:
```python
def return_object()->object:
    return int(30)
```

In [33]:
```python
return_object()
```

Out[33]: 30

In [34]:
```python
def nested_function()->callable:
    def hello(name:str)->str:
        return 'hello ' + name
    return hello
```

In [35]:
```python
nested_function()('ZL')
```

Out[35]: 'hello ZL'

In [36]:
```python
## 装饰器
import time

def timer(func:callable)->callable:
    def timed(*args, **kwargs):
        b = time.perf_counter_ns()
```

```
        r = func(*args, **kwargs)
        e = time.perf_counter_ns()
        print(f'time lapsed(ns) : {e-b:,.2f}')
        return r
    return timed
```

In [37]:
```
## 迭代器

def numbers(n:int=10)->int:
    for i in range(n):
        yield i
```

In [38]:
```
my_number = numbers(5)
next(my_number)
```

Out[38]: 0

In [39]:
```
next(my_number)
```

Out[39]: 1

# 3. 达 Statement and expression

## What

1. Arithmetic: + - * / **
2. Relational: = != > >= < <=
3. Logical: not and or
4. Assignment: =

## Why

模拟数学表达

## How

```
    a:int = 42; b:int = 69

    a + b
```

In [40]:
```
a, b = 42, 69
```

In [41]:
```
a + b
```

Out[41]: 111

In [42]:
```
a - b
```

Out[42]: −27

In [43]:
```python
a * b
```

Out[43]: 2898

In [44]:
```python
a / b
```

Out[44]: 0.6086956521739131

In [45]:
```python
a ** b
```

Out[45]: 100972018328803555738757908632148332268961863698723269942503985703768774336860
09543845316266007917815719968899072

In [46]:
```python
a % b
```

Out[46]: 42

In [47]:
```python
a == b
```

Out[47]: False

In [48]:
```python
a != b
```

Out[48]: True

In [49]:
```python
a > b
```

Out[49]: False

In [50]:
```python
a >= b
```

Out[50]: False

In [51]:
```python
a < b
```

Out[51]: True

In [52]:
```python
a <= b
```

Out[52]: True

In [53]:
```python
a and b
```

Out[53]: 69

In [54]:
```python
a or b
```

Out[54]: 42

In [55]:
```python
not a
```

Out[55]:  False

# 4. 控 Control flow

## What

条件语句，跟自然语言的概念一样。

如果天气预报说今天要打雷下雨⛈️，那就要带🌂

## Why

模拟自然语言

## How

```
if condi:
    ...
elif condi:
    ...
else:
    ...

try:
    ...
except Exception:
    ...
finally:
    ...
```

In [56]:
```python
x:int = 42

if x < 20:
    print(f'{x} is less than 20')
elif x == 20:
    print(f'{x} is equal to 20')
else:
    print(f'{x} is greater than 20')
```

42 is greater than 20

In [57]:
```python
try:
    rv = x / 0
except Exception as e:
    print(e)
finally:
    print(x)
```

division by zero
42

# 5. 流 Loop

## What

循环♻️

for...

while...

## Why

重复的工作让程序自动做

## How

```
for i in range(1, 11, 2):
    print(i)

i:int = 10

while i > 0:
    print(i)
    i -= 2
```

In [45]:
```python
for i in range(1, 11, 2):
    print(i, end=' ')
#     print(i)
```

1 3 5 7 9

In [59]:
```python
i:int = 9

while i > 0:
    print(i, end=' ')
    i = i - 2
```

9 7 5 3 1

# 6. 类 Class

## What

模拟现实中的某类东西。譬如：狗🐶，猫🐱，花🌹，人类💃，衣服👔

## Why

这类东西都是独立的。有自己的系统。

## How

```
class Dog:
    def __init__(self, name, age, sex):
        self._name = name
        self._age  = age
        self._sex  = sex
```

In [46]:
```python
class Dog:
    def __init__(self, name, age, sex):
        self._name = name
        self._age  = age
        self._sex  = sex

    def __str__(self)->str:
        return f'Dog: name is {self._name}, age is {self._age}, sex is {self._
```

In [47]:
```python
d1 = Dog('dog1', 3, 'male')
d2 = Dog('dog2', 5, 'female')
d3 = Dog('dog3', 1, 'unknown')
```

In [49]:
```python
print(d1)
print(d2)
print(d3)
```

```
Dog: name is dog1, age is 3, sex is male
Dog: name is dog2, age is 5, sex is female
Dog: name is dog3, age is 1, sex is unknown
```

In [50]:
```python
class SpottedDog(Dog):
    _spotted:bool = True

    def __init__(self, name, age, sex, spotted=True):
        self._spotted = spotted
        super().__init__(name, age, sex)

    def __str__(self)->str:
        if self._spotted:
            return f'Spotted Dog: name is {self._name}, age is {self._age}, s
        else:
            return f'Dog: name is {self._name}, age is {self._age}, sex is {s
```

In [51]:
```python
sd1 = SpottedDog('max', 4, 'female')
sd2 = SpottedDog('puppy', 5, 'male', Falsse)
```

In [52]:
```python
print(sd1)
print(sd2)
```

```
Spotted Dog: name is max, age is 4, sex is female
Dog: name is puppy, age is 5, sex is male
```

# 7. 数据结构

## What

模拟现实中的大型容器。譬如：箱子，衣柜，集装箱，手提箱，背包

## Why

我们可以批量地处理大型容器里的物品

## How

```
numbers:list = [1, 2, 3]

workdays:tuple = ('Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat',
'Sun')

members:dict = {
    'name' : 'ZL',
    'age' : 99,
    'sex' : 'male'
}
```

In [53]:
```python
numbers:list = [1, 2, 3]

workdays:tuple = ('Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun')

members:dict = {
    'name' : 'ZL',
    'age' : 99,
    'sex' : 'male'
}
```

In [54]:
```python
numbers[0]
```

Out[54]: 1

In [55]:
```python
workdays[6]
```

Out[55]: 'Sun'

In [56]:
```python
members['age']
```

Out[56]: 99

In [57]:
```python
dogs = [d1, d2, d3]
for dog in dogs:
    print(dog)
```

```
Dog: name is dog1, age is 3, sex is male
Dog: name is dog2, age is 5, sex is female
Dog: name is dog3, age is 1, sex is unknown
```

In [58]:
```python
workdays:tuple = ('Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun')

for workday in workdays:
    print(workday)
```

```
Mon
Tue
Wed
Thu
Fri
Sat
Sun
```

In [60]:
```python
members:dict = {
    'name' : 'ZL',
    'age' : 99,
    'sex' : 'male',
    'pet' : Dog('meow', 3, 'female')
}

for k, v in members.items():
    print(k, v)
```

```
name ZL
age 99
sex male
pet Dog: name is meow, age is 3, sex is female
```

In [63]:
```python
class Cafe:
    def __init__(self, cup, label, lid, suger, water):
        self.cup = cup
        self.label = label
        self.lid = lid
        self.suger = suger
        self.water = water

    def __str__(self):
        return f'Cafe: {self.cup}'
```

In [64]:
```python
cafe1 = Cafe('Middle', 'McCafe', True, 'suger', 'water')
```

In [65]:
```python
print(cafe1)
```

```
Cafe: Middle
```

In [66]:
```python
class Latte(Cafe):
    _flavor:str = 'latte'
```

In [67]:
```python
latte1 = Latte('Big', 'McCafe', True, 'no suger', 'water')
```

In [70]:
```python
print(latte1, latte1._flavor)
```

```
Cafe: Big latte
```

In [71]:
```python
class MilkCafe(Cafe):
    flavor:str = 'Milk'
```

In [72]:
```python
mc = MilkCafe('Large', 'McCafe', False, 'suger', 'water')
```

In [73]:
```python
print(mc)
```

```
Cafe: Large
```

In [74]:
```python
print(mc.flavor)
```

```
Milk
```