python内置函数

```
# abs()            dict()          help()          min()          setattr()
# all()            dir()           hex()           next()         silce()
# any()            divmod()        id()            object()       sorted()
# ascii()          enumerate()     input()         oct()          staticmethod()
# bin()            eval()          int()           open()         str()
# bool()           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()
# delattr()        hash()          memoryview()    set()
```

```
# print()       input()

# abs()         divmod()        round()         pow()
# sum()         min()           max()

# str()         int()           float()         list()
# dict()        tuple()         set()           bool()

# format()      repr()

# len()         silce()         sorted()        reversed()          filter()
# all()         any()           iter()          next()              range()
# enumerate()   zip()           map()

# bin()         oct()           hex()
# bytes()       bytearray()     ord()           chr()

# super()       property()      classmethod()   staticmethod()
# callable()    isinstance()    issubclass()

# dir()         hasattr()       getattr()       setattr()       delattr()
# __import__()  eval()          exec()

# id()          type()          open()          hash()
# globals()     locals()        vars()
# ascii()       object()        frozenset()

# memoryview()  complex()       compile()
```

文件下载地址：https://active.clewm.net/FrcyFA

入门函数

```
# print()       input()
```

数学运算

```
# abs()         divmod()        round()         pow()
# sum()         min()           max()
print(abs(-2))  # 返回绝对值 >>>2
print(divmod(20,3)) # 返回商和余数    >>>(6,2)
print(round(4.50))   # 四舍五入有    >>>4
print(round(4.51))  # 四舍五入有 >>>5
print(pow(10,2,3))  # 求10的2次幂>>>100 .如果有三个参数. 则求完次幂后对第三个数取余 >>>1
print(sum([1,2,3,4,5,6,7,8,9,10]))  # 求和    >>>55
print(min(5,3,9,12,7,2))  #求最小值 >>>2
print(max(7,3,15,9,4,13))  #求最大值    >>>15
```

数据类型

```
# str()         int()           float()         list()
# dict()        tuple()         set()           bool()
```

字符串

```
s = "hello world!"
print(format(s, "^20"))  #中对齐   >>>    hello world!
print(format(s, "<20"))  #左对齐   >>>hello world!
print(format(s, ">20"))  #右对齐   >>>          hello world!

# 右对齐，横杠填满20个字符，并加上前为分隔符
print('{:->20,}'.format(1303544500.666))    # >>>---1,303,544,500.666

print(format(3, 'b' ))    # 3转二进制>>>11
print(format(97, 'c' ))   # 97转换成unicode字符>>>a
print(format(11, 'd' ))   # 11转十进制>>>11
print(format(11, 'o' ))   # 11转八进制 >>>13
print(format(11, 'x' ))   # 11转十六进制(小写字母) >>>b
print(format(11, 'X' ))   # 11转十六进制(大写字母) >>>B
print(format(11, 'n' ))   # 和d一样 >>>11
print(format(11))         # 和d一样 >>>11
print(format(123456789, 'e' ))      # 科学计数法. 默认保留6位小数 >>>1.234568e+08
print(format(123456789, '0.2e' ))   # 科学计数法. 保留2位小数(小写) >>>1.23e+08
print(format(123456789, '0.2E' ))   # 科学计数法. 保留2位小数(大写) >>>1.23E+08
print(format(1.23456789, 'f' ))     # 小数点计数法. 保留6位小数 >>>1.234568
print(format(1.23456789, '0.2f' ))  # 小数点计数法. 保留2位小数 >>>1.23
print(format(1.23456789, '0.10f'))  # 小数点计数法. 保留10位小数 >>>1.2345678900
print(format(1.23456789e+3, 'F'))   # 小数点计数法. 很大的时候输出INF >>>1234.567890

print(repr(f"今天\n吃了{3}顿\t饭"))   # 保留引号，原样输出 >>>'今天\n吃了3顿\t饭'
```

序列迭代器函数

```
# len()         silce()         sorted()        reversed()          filter()
# all()         any()           iter()          next()              range()
# enumerate()   zip()           map()

print(len("hello world"))      #>>>11

lst = [1, 2, 3, 4, 5, 6, 7]
print(lst[1:3:1])  #[2,3]
s = slice(1, 3, 1)  #  切片用的
print(lst[s])  #[2,3]


lst = [5,7,6,12,1,13,9,18,5]
lst.sort()  # sort是list里面的一个方法
print(lst)  # >>>[1, 5, 5, 6, 7, 9, 12, 13, 18]
ll = sorted(lst) # 内置排序函数，默认升序，返回一个新列表
print(ll)  # >>>[1, 5, 5, 6, 7, 9, 12, 13, 18]
l2 = sorted(lst,reverse=True)  #降序
print(l2)  # >>>[18, 13, 12, 9, 7, 6, 5, 5, 1]

#根据字符串长度给列表排序
lst = ['one', 'two', 'three', 'four', 'five', 'six']
l1 = sorted(lst, key=lambda s:len(s))
print(l1)  # >>>['one', 'two', 'six', 'four', 'five', 'three']


lst = "你好啊"
it = reversed(lst)   #将一个序列翻转，不会改变原值，返回翻转序列的迭代器
li = list(it)
print(li)  # >>>['啊', '好', '你']
li.reverse()    # list里面的reverse方法也有类似的效果
print(li)   # >>>['你', '好', '啊']


# filter()  # 用于过滤序列，过滤掉不符合条件的元素
it = filter(lambda n: n % 2 == 0, [1,11,55,6,5,73,84,99])
print(it)               # >>> <filter object at 0x000001DAFE19F848>
for x in it:
    print(x)
>>>6
>>>84


print(all([1, 'hello', True, 0]))   #可迭代对象中有一个为False，结果就是False，
>>>False
print(all([]))  # 如果可迭代对象为空，返回True  >>>True


print(any([0,False,1,'good']))  # 可迭代对象中有一个是True，结果就是True  >>>True
print(any([]))  # 如果可迭代对象为空，返回Fasle >>>False
```

```
it = iter(["Python", "Linux", "go"])    # 用来生成迭代器
>>> <list_iterator object at 0x000001DAFE19F948>

next(it)    # >>> Python    返回迭代器下一个元素
next(it)    # >>> Linux     返回迭代器下一个元素

range(1,10)     # 返回数字序列


# enumerate() 枚举
lst = ['one','two','three','four','five']
for index, item in enumerate(lst,1):    # 把索引和元素一起获取,索引默认从0开始. 可以更改为1
    print(index, item)
>>>1 one
>>>2 two
>>>3 three
>>>4 four
>>>5 five


# zip() 拉链
"""
zip函数用于将可迭代的对象作为参数，将对象中对应的元素打包成一个元组，然后返回由这些元组组成的迭代器. 如果各个可迭代对象的元素长度不一致，则返回迭代的长度与最短的可迭代相同
"""
lst1 = [1, 2, 3, 4, 5, 6]
lst2 = ['醉乡民谣', '驴得水', '放牛班的春天', '美丽人生', '辩护人', '被嫌弃的松子的一生']
lst3 = ['美国', '中国', '法国', '意大利', '韩国', '日本']

print(zip(lst1, lst1, lst3))  # 返回一个迭代器
>>> <zip object at 0x00000256CA6C7A88>

for el in zip(lst1, lst2, lst3):
    print(el)
>>>(1, '醉乡民谣', '美国')
>>>(2, '驴得水', '中国')
>>>(3, '放牛班的春天', '法国')
>>>(4, '美丽人生', '意大利')
>>>(5, '辩护人', '韩国')
>>>(6, '被嫌弃的松子的一生', '日本')


# map() 根据提供的函数对指定序列做映射。
map(lambda x: x ** 2, [1, 2, 3, 4, 5])  # 计算平方数
>>> [1, 4, 9, 16, 25]
```

进制与编码

```
# bin()         oct()           hex()
# bytes()       bytearray()     ord()           chr()

print(bin(10))  # 将给的参数转换成二进制    >>>0b1010
print(oct(10))  # 将给的参数转换成八进制    >>>0o12
print(hex(10))  # 将给的参数转换成十六进制   >>>0xa

bs = bytes("我喜欢你", encoding="utf-8")    # 把字符串转化成bytes类型
print(bs)
>>>b'\xe6\x88\x91\xe5\x96\x9c\xe6\xac\xa2\xe4\xbd\xa0'

bytearray()    # 返回一个新字节数组. 这个数组的元素是可变的, 并且每个元素的值得范围是
[0,256)
ret = bytearray("fei" ,encoding ='utf-8')
print(ret)
>>>bytearray(b'fei')
print(ret[0])
>>>102
ret[1]=102  #把102的位置f赋值给ret[1]
print(ret)
>>>bytearray(b'ffi')

print(ord('a'))  # 字母a在编码表中的码位   >>>97
print(ord('飞'))  # '飞'字在编码表中的位置 >>>39134

print(chr(65))  # 已知码位,求字符是什么    >>>A
print(chr(39134))  # >>>飞
```

类与对象相关

```
# super()       property()      classmethod()   staticmethod()
# callable()    isinstance()    issubclass()

# super() 用于调用父类的方法
# property() 装饰器，用于把类方法转换为类属性
# classmethod() 装饰器，类绑定方法
# staticmethod() 装饰器，非绑定方法

#callable() # 判断一个对象是否可以加括号调用
def func():
    pass
print(callable(func))   # >>>True
class Test():
    pass
print(callable(Test))   # >>>True

# isinstance() # 判断一个对象，是否是某一个类的实例
print(isinstance('abc', str))   # >>>True
```

```
# issubclass()  判断一个类是否是另一个类的子类
class Human():
    pass

class Chinese(Human):
    name = "张大仙"
    age = 73

res = issubclass(Chinese,Human)            #判断Chinese是否是Human的子类
print(res)
>>>True
```

反射相关

```
# dir()         hasattr()       getattr()       setattr()       delattr()
# __import__()  eval()      exec()

dir(obj)    # 查看一个对象下可以点出来的所有属性
hasattr(obj, 'age')  # 查看对象是否有age属性
getattr(obj, 'age') # 获取对象的age属性
setattr(obj, 'age',  18) # 给对象的age属性设置值
delattr(obj, 'age') # 删除对象的age属性

# __import__()  # 用字符串方式导入模块
time = __import__('time')
print(time.time())


# eval()    # 执行字符串类型的代码（适用于表达式执行），并返回最终结果
res = eval('1+2+b', {'b': 3}, {'b': 4})
print(res)
>>>7



# exec()    # 执行字符串类型的代码（适用于代码块执行），把产生的局部名称空间给y，没有返回值
y = {}
res = exec('a=1+2+b', {'b': 3}, y)
print(res, y)
>>>None {'a': 6}

y = {'b': 4}
res = exec('a=1+2+b', {'b': 3}, y)
print(res, y)
>>>None {'b': 4, 'a': 7}
```

其他

```
# id()          type()      # open()        # hash()
# globals()     locals()    vars()
# ascii()       object()    frozenset()

id()    # 在CPython中，用于获取对象的内存地址。
type()  # 返回对象的类型
open()  # 用户打开文件，创建一个 file 对象
hash()  # 哈希加密，这个内置函数一般不用，都用hashlib模块

# globals() 以字典形式返回当前位置的全部全局变量
name = '张大仙'
print(globals())
>>>{'__name__': '__main__', '__doc__': None, '__package__': None, '__loader__':
<_frozen_importlib_external.SourceFileLoader object at 0x0000024F6E0C0888>,
'__spec__': None, '__annotations__': {}, '__builtins__': <module 'builtins'
(built-in)>, '__file__': 'D:/teach/进阶篇/settings.py', '__cached__': None,
'name': '张大仙'}

#locals()   以字典形式返回当前位置的全部局部变量，如果直接在全局打印，结果会和globals()一样
def func(n):  # 两个局部变量：n、z
    z = 1
    print(locals())

func(2)
>>>{'n': 4, 'z': 1}

#vars() 返回对象所有属性（等同于__dict__）。如果不传参数，就打印当前调用位置所有属性，等同于
locals()。
# 但和hasattr不一样，hasattr是看对象可以点出来哪些属性，包括对象类里面的属性，父类的属性，都是
可以点出来的
class Human:
    star = 'earth'
    def __init__(self, name, age):
        self.name = name
        self.age = age
print(vars(Human))
>>>{'__module__': '__main__', 'star': 'earth', '__init__': <function
Human.__init__ at 0x000001321EFCDC18>, '__dict__': <attribute '__dict__' of
'Human' objects>, '__weakref__': <attribute '__weakref__' of 'Human' objects>,
'__doc__': None}

obj = Human('张大仙', 18)
print(vars(obj))
>>>{'name': '张大仙', 'age': 18}


# ascii()     # 将所有非 ascii 字符替换为转义字符，返回的结果会保留引号，与repr类似
ascii('hello 你好')
>>>'hello \\u4f60\\u597d'

# frozenset()   # 用set()定义的集合为可变集合。而它返回的是一个冻结的集合，该集合不能再添加
或删除任何元素，相当于不可变集合。
s = frozenset('123')
```

```
print(s)
>>>frozenset({'2', '1', '3'})

# object()    # 调用它会返回一个空对象，但一般都不会调用它，都是用它作为类的基类使用。

# memoryview()  # 返回给定参数的内存视图对象，不用掌握
# complex()       # 返回一个复数，  不用掌握
# compile() # 将一个字符串编译为字节代码，不用掌握
```