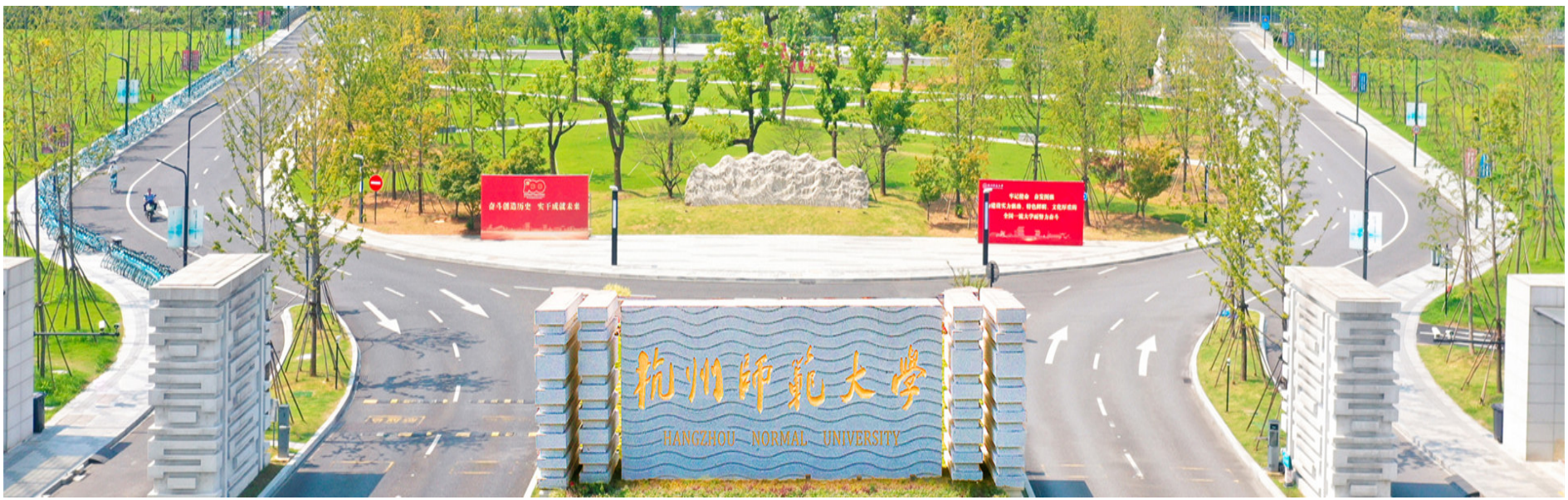
第五讲 - Python 函数

张建章

阿里巴巴商学院

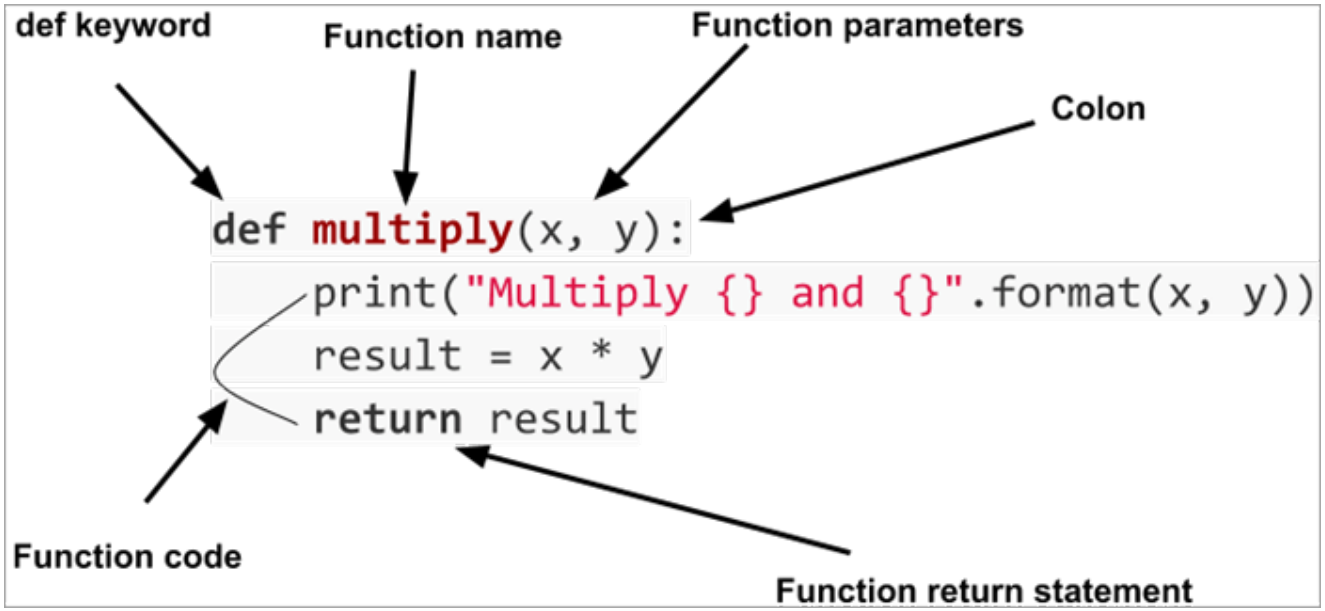
杭州师范大学

2022-09



目录 I

函数: 为实现一个操作或特定功能而集合在一起的语句集，避免代码复制带来的错误或漏洞，不仅可以实现代码的复用，还可以保证代码的一致性。



一个完整的函数由如上图所示的几部分组成：`def` 关键字 (def keyword)，函数名 (function name)，函数参数 (function parameters)，冒号 `:` (colon)，函数体 (function code)。

```
def multiply(x,y):
  print("Multiply {} and {}".format(x, y))
  result = x * y
  return result
```

注意：函数体中的`return`语句不是必需的，如果不写，调用函数后返回`None`，如果所定义的函数不需要参数，可以不写参数，但是括号必需写，看下面例子：

```
def color_print():
  for i in range(91,96):
    print("\033["+str(i)+"m"+"zjzhang"+"\033[0m")

result = color_print()
if result == None:
  print('The call of Function color_print returns None')
```

函数调用

```
# 调用multiply函数，忽略返回值
multiply(3, 5)

# 调用multiply函数，并将函数的返回值复制给变量r
r = multiply(3, 5)

print(r) # 15

# 调用color_print函数，忽略返回值None
color_print()

# 调用color_print函数，将返回值赋值给变量r
# 对于没有参数的函数，这种调用方式没什么意义
r = color_print()
print(r)
```

注意: ① 函数调用可以将返回值赋值给其他变量，也可以忽略返回值；② 函数调用必须写括号，无论是否需要写参数。

局部变量: 在函数体内定义的变量，只能在函数内部被访问。

全局变量: 在所有函数之外创建的变量，可以在程序的任意位置访问。

```
# 此处的x, y为全局变量
x, y = 3, 5

def myfunct():
  # 此处的x, y为局部变量
  x, y = 8, 9
  result = x * y

p = myfunct() # 函数调用
# 使用全局变量做计算并将计算结果赋值给全局变量z
z = x + y
print(z) # 8, 不是17
print(p) # None
```

由上例可见，局部变量和全局变量可以同名，但是在函数内部定义的局部变量不能在函数外访问。

形式参数与实际参数

```python
# 定义函数
def multiply(x,y):
  print("Multiply {} and {}".format(x, y))
  result = x * y
  return result

# 调用函数
p, q = 8, 9
multiply(p, q)
```

在上例中，定义函数时写的参数 x 和 y 是形式参数 (形参)，调用函数时传递给函数的参数 p 和 q (或者说 8 和 9) 叫做实际参数 (实参)。

Python 的参数传递

```
# 定义函数
def multiply(x,y):
  print("Multiply {} and {}".format(x, y))
  result = x * y
  return result

# 调用函数

p, q = 8, 9
# 这是引用传递
multiply(p, q)
# 这是值引用
multiply(5, 6)
```

注意：引用传递的本质是将变量指向的数据对象传递到函数中，上例中，变量 p 和 q 指向的数据对象为整数 8 和 9。

```python
# 定义函数
def my_double(my_obj):
  my_obj *= 2
  return my_obj

a = '000'
b = [1,2]
# 函数调用
my_double(a)
my_double(b)
print(a) # 000
print(b) # [1,2,1,2]
```

上面例子中，变量 a 和变量 b 分别指向不可变对象 (字符串) 和可变对象 (列表)，因此，在函数调用后，变量 b 的值发生了变化，变量 a 的值不变。所以，**当进行引用传递时，要注意，如果变量指向的是可变对象，函数体的代码是可以改变变量值的。**

默认参数

定义函数时，可以为参数指定默认值。

如果在调用函数时，未指定参数，则使用参数的默认值作为参数值运行函数；

如果参数不存在默认值，并且调用时也没指定参数值，则程序出错。

```
def my_power(base, exponent=2):
  return base ** exponent

print(my_power(3)) # 9
print(my_power()) # Error
```

上例中定义的函数 `my_power`，包含两个参数 base 和 exponent，参数 base 没有默认值，在调用函数时必须为其指定值，参数 exponent 有默认值，为 2，调用函数时，如不指定其值，则使用 2 作为其值运行函数。

定义函数时，没有默认值的参数在前，具有默认值的参数在后，否则，程序出错 (SyntaxError)。

```
# Correct
def my_power(base, exponent=2):
  return base ** exponent

# Error
def my_power(exponent=2, base):
  return base ** exponent
```

Python 不支持函数重载。

函数重载 (方法重载): 是某些编程语言 (如 C++、C#、Java、Swift、Kotlin 等) 具有的一项特性，该特性允许创建多个具有不同实现的同名函数。对重载函数的调用会运行其适用于调用上下文的具体实现，即允许一个函数调用根据上下文执行不同的任务。

按位置传参和按关键字传参

按位置传参: 按照函数定义时的顺序进行参数传递。

按关键字传参: 按照 `name=value` 的格式进行参数传递。

```
# my_sum的三个参数均有默认值
def my_sum(a = 1, b = 2, c = 3):
  return a**0 + b**1 + c**2

# 按位置传参
my_sum(7, 8, 9)
# 按关键字传参
my_sum(c=7, a=8, b=9)
# 按位置传参和按关键字传参混合使用
my_sum(7, 8, c=9)

my_sum(7, b=8, 9) # Error
```

注意：按位置传参和按关键字传参混合使用时，位置位置参数在前，关键字参数在后，否则，程序出错 (SyntaxError)。

```
# 按关键字传参，对有默认值的参数使用其默认值
my_sum(a=9)

# 按位置传参，对有默认值的参数使用其默认值
my_sum(7)

# 按位置传参和按关键字传参混合使用
my_sum(7, c=8)

# 按位置传参和按关键字传参混合使用，但是给参数a传递了多个值
my_sum(7, a=8) # Error
```

可变长度参数

注意：在定义函数时，不固定参数的数量，调用时也可以使用不同个数的参数，可使用两种参数名来实现：`*args` 和 `**kwargs`，分别对应参数元组 (按位置传参) 和参数字典 (按关键字传参)。

```
def hello_args(para1, *args):
  print("para1 :", para1)
  print("type(args):", type(args))
  print("args :", args)
  for idx,arg in enumerate(args):
    print("args{}:".format(idx+1), arg)

hello_args() # Error，必须要为para1指定参数值，因为它没有默认值
hello_args('hello') # 按位置传参，没有为*args传参
hello_args('hello', 'this', 'is', 'mc.zhang') # 按位置传参
```

上例中，`*args` 用来定义个数不确定的参数**元组**，在函数体中可以通过 `for` 循环和索引访问参数元组，对于 `*args`，可以传递 0 或多个参数。

```python
def hello_args(para1, *args):
  print("para1 :", para1)
  print("type(args):", type(args))
  print("args :", args)
  for idx,arg in enumerate(args):
    print("args{}:".format(idx+1), arg)

args_list = ['this', 'is', 'mc.zhang']
# 先序列(列表)解封，再按位置传参
hello_args(*args_list)
# 先序列(列表)解封，再按位置传参
hello_args('hello', *args_list)

# 先序列(列表)解封，再混合使用按关键字传参和按位置传参，不行
# 这样就违背了“位置参数在前，关键字参数在后”的规则
hello_args(para1 = 'hello', *args_list) # Error
```

```python
def hello_kwargs(para1, **kwargs):
  print("para1 :", para1)
  print("type(kwargs):", type(kwargs))
  print("kwargs:", kwargs)
  for key, value in kwargs.items():
    print("{0} = {1}".format(key, value))

hello_kwargs() # Error，必须要为para1指定参数值，因为它没有默认值
hello_kwargs('hello') # 按位置传参，没有为*kwargs传参
# 按位置传参和按关键字传参混合
hello_kwargs("hello", kwarg_1='this', kwarg_2='is',
↪  kwarg_3='mc.zhang')
```

上例中，`*kwargs` 用来定义个数不确定的参数**字典**，在函数体中可以通过 `for` 循环和关键字 (key) 访问参数字典，对于 `*kwargs`，可以传递 0 或多个参数。

```python
def hello_kwargs(para1, **kwargs):
  print("para1 :", para1)
  print("type(kwargs):", type(kwargs))
  print("kwargs:", kwargs)
  for key, value in kwargs.items():
    print("{0} = {1}".format(key, value))

kwargs_dict = {'kwarg_1': 'this', 'kwarg_2': 'is', 'kwarg_3':
↪  'mc.zhang'}
# 先字典解封，再按位置传参和按关键字传参混合
hello_kwargs("hello", **kwargs_dict)
# 按位置传参，Error，对于**kwargs，只能按关键字传参
hello_kwargs('hello', 'this', 'is', 'mc.zhang')
# 先字典解封，再按关键字传参
hello_kwargs(para1 = "hello", **kwargs_dict)
hello_kwargs(**kwargs_dict, para1 = "hello")
# Error，违背了“位置参数在前，关键字参数在后”的规则
hello_kwargs(**kwargs_dict, "hello")
```

本小节通过一系列函数实例展现函数的功能封装性。

题目：给定一个身份证号，验证其是否合法：① 总体校验；② 校验地区；③ 校验日期。

```
# 校验字符和长度
def verify_char_length(ids):
  if len(ids) != 18:
    return False
  if not ids[:-1].isdigit():
    return False
  if ids[-1] not in '0123456789X':
    return False
  return True
```

```python
# 校验第18位是否正确
def verify_last_num(ids):
  ids_17 = ids[:-1]
  weights = [7, 9, 10, 5, 8, 4, 2, 1, 6, 3, 7, 9, 10, 5, 8, 4, 2]
  S = sum([int(num)*weight for num,weight in
  ↪  zip(list(ids_17),weights)])
  T = S % 11
  R = (12 - T) % 11
  if R == 10:
    last_num = 'X'
  else:
    last_num = R
  if ids[-1] == str(last_num):
    return True
  else:
    return False
```

```python
# 校验前六位地区编码是否正确
def verify_area(ids):
  import _locale
  _locale._getdefaultlocale = (lambda *args: ['zh_CN', 'utf8'])
  with open('./area_dict.json') as f:
    area_dict = eval(f.read())
  if ids[:6] not in area_dict.keys():
    return False
  else:
    return True

# 校验闰年
def is_leap_year(year):
  if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    return True
  else:
    return False
```

```python
def verify_date(ids):
  year = int(ids[6:10])
  if year > 2022 or year < 1900:
    return False
  month = int(ids[10:12])
  if month > 12 or month < 1:
    return False
  day = int(ids[12:14])
  if month in [1,3,5,7,8,10,12]:
    if day > 31 or day < 1:
      return False
  elif month in [4,6,9,11]:
    if day > 30 or day < 1:
      return False
  else:
    if is_leap_year(year):
      if day > 29 or day < 1:
        return False
    else:
      if day > 28 or day < 1:
        return False
  return True
```

```
def verify_id(ids):
  if all([verify_char_length(ids),verify_last_num(ids),verify_ar」
  ↪  ea(ids),verify_date(ids)]):
    print("VALID")
  else:
    print("INVALID")

# https://www.dute.org/fake-id-card-number
verify_id(ids = "11011319870212l018") # VALID
verify_id(ids = "11011319870312l018") # INVALID
```

未完待续