Введение в Data Science

Этап 1. Язык программирования Python

С. В. Сергеенко Д. В. Бирюкова

ООО «ЛАЦИТ — Лаборатория цифровых технологий»

2021

Функции в Python

Определение функции

- является составным оператором в Python;
- состоит из указаний декораторов, ключевого слова def, имени функции и списка параметров, за которым идёт тело функции;
- декораторы применяют к функции заранее подготовленные модификации;
- значения по умолчанию для функции вычисляются один раз при её определении;
- определение функции не влечёт немедленного выполнения её тела,
- для выполнения тела функции, её надо вызвать.

Синтаксис определения функции

```
@decorator_expression1
@decorator2
def funcname(arg1 : int ,/     # onlypos
             ,arg2 = 4,*args  # any (except *)
             ,kwarg,**kw): # onlykw (except **)
  "Documentation of funcname"
  print("arg1:", arg1)
  print("arg2:", arg2); print("args:", args)
  print("kwarg:", kwarg); print("kw:", kw)
  return NotImplemented
funcname(1,2,3,4 # pos
         ,arg1=5,kwarg=6,x=7) # keyword
print(funcname.__name__,funcname.t)
```

Безымянные функции

- другое название лямбда-выражение (λ-выражение);
- являются упрощенным вариантом функции, тело которой содержит только одно выражение, которое определяет возвращаемое значение;
- создаются выражением, значением которого является функция.

Примеры

```
print(lambda *p, **k: print(p, k))
print((lambda *p, **k: print(p, k))(1,2,3,a=4))
```

Функция также объект

Функция — объект

- функция в Python, как и всё остальное, представляется объектом;
- функцию можно присваивать всему, чему можно присваивать объект (имени, элементу списка);
- функцию можно передавать как параметр в функцию;
- функцию можно возвращать из функции.

Примеры

```
1 def f(g):
    print(g)
    return g
4 f(f)
```

Примеры (прод.)

```
def decorator_expressio1(f):
    def x(*p, **k):
        print(x)
        print(p)
        print(k)
        print(f)
        return f(*p, **k)
    x.t = 10
    return x
```

Примеры (прод.)

```
def decorator2(f):
    print(f,f.__name__)
    print(f.__code__)
    print(f.__doc__)
    print(dir(f))
    print(f.__annotations__)
    print(f.__defaults__)
    print(f.__kwdefaults__)
    help(f)
    return f
```

Функции с фиксированным/переменным количеством параметров

Виды аргументов

- можно выделить два вида аргументов: позиционные и именованные;
- позиционные аргументы связываются с параметрами на основе их позиции в списке аргументов;
- именованные аргументы связываются с параметрами на основе предоставленного имени;

Распаковка аргументов

Если внутри списка аргументов в вызове функции

- перед выражением, означающим некоторую последовательность, стоит символ *, то каждый элемент последовательности добавляется как отдельный позиционный аргумент вместо этого выражения;
- перед выражением, означающим некоторое отображение (словарь), стоит **, то каждая пара ключ-значение добавляется как именованный аргумент (ключ определяет имя) вместо этого выражения.

Виды параметров

- по тому, с какими аргументами могут связываться, параметры делим на исключительно позиционные (onlypos), произвольные (any) и исключительно именованные (onlykw);
- произвольные могут быть связываны с позиционными и именованными аргументами;
- исключительно позиционные — только с позиционными аргументами;
- исключительно именованные — только с именованными аргументами.

Специальные параметры

- Если после *, отделяющей произвольные параметры от исключительно именованных указано имя, то оно будет связано с кортежем позиционных аргументов, для которых нет соответствующих параметров.
- Если в списке параметров присутствует тот, перед которым стоит **, то этот параметр будет связан со словарём именованных аргументов, для которых нет соответствующих параметров.

Связывание аргументов с параметрами

- при сопоставлении аргументы рассматриваются поочерёдно слева направо;
- связывание имён параметров со значениями проходит в три этапа:
  1. связывание позиционных аргументов;
  2. связывание именованных аргументов;
  3. связывание со значениями по умолчанию.

Связывание аргументов с параметрами

Связывание позиционных аргументов

- сначала происходит связывание позиционных аргументов с параметрами (кроме исключительно именованных) — каждый аргумент связывается с самым левым допустимым несвязанным параметром;
- избыточные позиционные аргументы помещаются в кортеж, связываемый со специальным параметром *, если он есть;

Связывание аргументов с параметрами

Связывание именованных аргументов

- затем, каждый именованный аргумент связывается с параметром с совпадающим именем (исключительно позиционные параметры не рассматриваются);
- именованные аргументы, для которых нет соответствующего параметра, заносятся в словарь, связываемый со специальным параметром **, если он есть;

Связывание аргументов с параметрами

Связывание со значениями по умолчанию

- в конце, параметры, не связанные с аргументами, связываются со своими значениями по умолчанию, если они есть.

Ошибки вызова

Можно выделить следующие ошибки при вызове:

- наличие избыточного числа позиционных аргументов (если нет параметра *);
- наличие именованного аргумента без соответствующего параметра (если нет **);
- соответствие параметра как позиционному, так и именованному аргументу;
- наличие нескольких одноименных параметров;
- наличие параметра без заданного значения по умолчанию, не соответствующего никакому аргументу.

Особенности значений по умолчанию в Python

Как было сказано ранее, значения по умолчанию вычисляются при определении функции. Это может приводить к неожиданным последствиям, если объекты для значений по умолчанию являются мутабельными.

Особенности значений по умолчанию в Python

Примеры

Изменение значение по умолчанию

```
def expanded_list(L = []):
    L.append(3); return L
print(expanded_list()); print(expanded_list())
```

Значение по умолчанию неизменно

```
def expanded_list2(L = None):
    if L is None: L = []
    L.append(3); return L
print(expanded_list2()); print(expanded_list2())
```

Переменное количество аргументов

Переменное количество аргументов может быть у функции за счёт:

- отсутствия аргументов для параметров с указанным значением по умолчанию;
- использования специальных параметров получающих избыточные аргументы.

Ссылки

- Описание определения функции в справочнике по Python
- Пример работы с исключениями, неперехваченными в функции, где они возникли
- Урок по использованию аннотаций
- Поддержка проверки типов стандартной библиотекой
- Пример декорирования, передачи аргументов и использования атрибутов функции