Въведение в Python

Introducing Python

Flexible and powerful, Python was originally developed in the late 1980s at the National Research Institute for Mathematics and Computer Science by Guido van Rossum as a successor to the ABC language. Since its introduction, Python has grown in popularity thanks to what is seen as a clear and expressive syntax developed with a focus on ensuring that code is readable.

Python is a *high-level language*. This means that Python code is written in largely recognisable English, providing the Pi with commands in a manner that is quick to learn and easy to follow.

This is in marked contrast to *low-level languages*, like assembler, which are closer to how the computer "thinks" but almost impossible for a human to follow without experience.

The high-level nature and clear syntax of Python make it a valuable tool for anyone wh owants to learn to program. It is also the language that is recommended by the Raspberry Pi Foundation for those looking to progress from the simple Scratch (described in Chapter 10, "An Introduction to Scratch") to more "hands-on" programming.

Python is published under an open source licence and is freely available for Linux, OS X, and Windows computer systems. This cross-platform support means that software written using Python on the Pi can be used on computers running almost any other operating system as well—except where the program makes use of Pi-specific hardware such as the GPIO (general-purpose input/output) port. To find out how Python can be used to address this port, see Chapter 14, "The GPIO Port".

Python е интерпретируем, интерактивен, обектно-ориентиран език за програмиране, създаден от Гуидо ван Росум в началото на 90-те години. Кръстен е на телевизионното шоу на BBC *„Monty Python's Flying Circus"*.

История

Идеята за Python се заражда в края на 1980-те, като реалното осъществяване започва през декември 1989 г. от Гуидо ван Росум в CWI (Centrum Wiskunde & Informatica – международно признат изследователски институт по математика и компютърни науки, базиран в Амстердам, Холандия). Python имал за цел да се превърне в наследник на ABC (език за програмиране, от своя страна вдъхновен от SETL), който да бъде способен да обработва изключения и да е съвместим с операционната система Amoeba. Ван Росум е основният автор на Python, а неговата продължаваща централна роля в развитието на езика е ясно отразена в титлата, дадена му от Python общността – „пожизнен доброжелателен диктатор" (benevolent dictator for life (BDFL)).

Употреба

От 2003 г. насам, Python се класира в топ 10 на най-популярните езици за програмиране според TIOBE Programming Community Index. От септември 2015 г. той заема пета позиция. Обявен е за език за програмиране на годината през 2007 г. и 2010 г. Това е третият най-популярен език, чийто граматически синтаксис не е базиран предимно на C, както е при C++ и Objective-C. За сведение C# и Java имат само частично сходство на

синтаксиса със C, като например използването на къдрави скоби, и са по-близки помежду си отколкото със C.

Практиката доказва, че скриптовите езици като Python са по-продуктивни от конвенционалните такива (C и Java), що се отнася до разрешаването на програмни проблеми като манипулацията на низове и търсене в речник. Количеството използвана памет често е по-малко отколкото при Java и не много повече отколкото при C и C++.

Част от големите организации, които използват Python са Google, Yahoo!, CERN, NASA, както и някои по-малки като ILM и ITA.

Python може да се използва като скриптов език за уеб приложения, например чрез mod_wsgi за Apache. С уеб сървъра Gateway Interface, стандартният Приложно-програмен интерфейс се подобри, за да може да обслужва тези приложения. Софтуерните рамки за уеб приложения като Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle and Zope помагат на разработчиците при дизайна и поддръжката на по-сложни приложения. Pyjamas и IronPython могат да се използват при програмирането на клиентската страна (Client-side) на приложенията, базирани на Ajax. SQLAlchemy може да се използва при мапване на данни от сходни база данни. Twisted е софтуерна рамка за програмиране на комуникацията между компютрите и се използва от Dropbox.

Библиотеки като NumPy, SciPy и Matplotlib позволяват ефективното използване на Python за научни изчисления. Sage е математически софтуер, който е написан предимно на Python и обхваща много аспекти на математиката, включително алгебра, комбинаторика, теория на числата и др.

Python е внедрен успешно като скриптов език в редица софтуерни продукти, включително в такива използващи метода на крайните елементи като Abaqus, FreeCad, програми и инструменти за 3D анимации като 3ds Max, Blender, Cinema 4D, Lightwave, 2D програми за обработка на изображения като GIMP, Inkscape, Scribus and Paint Shop Pro. GNU Debugger (GDB) използва Python за визуализация на сложни структури като C++ контейнери за елементи. ESRI представя Python като най-добрия избор на скриптов език за ArcGIS. Също така се използа във видео игри и е единият от трите възможни езика за програмиране, приети от Google App Engine, другите два са Java и Go.

Като скритпов език с модулна архитектура, прост синтаксис и богат набор инструменти за обработка на текст, Python се използва при проекти, свързани с изкуствен интелект и компютърна обработка на човешки език (OEE).

Python се използва широко и в сферата на информационната сигурност.

По-голямата част от софтуера Sugar за One Laptop per Child XO, който се разработва в Sugar Labs, е написан на Python. Той е главен език за програмиране и в проекта Raspberry Pi, разработващ едноплаткови компютри. LibreOffice включва Python и възнамерява да замени Java с него.

Python Scripting Provider заема възлова позиция след излизането на версия 4.0 на 07.02.2013 г.

Структура и функционалност

„Python“ предлага добра структура и поддръжка за разработка на големи приложения. Той притежава вградени сложни типове данни като гъвкави масиви и речници, за които биха били необходими дни, за да се напишат ефикасно на C.

Python позволява разделянето на една програма на модули, които могат да се използват отново в други програми. Също така притежава голям набор от стандартни модули, които да се използват като основа на програмите. Съществуват и вградени модули, които обезпечават такива неща като файлов вход/изход (*I/O*), различни системни функции, сокети (*sockets*), програмни интерфейси към GUI (графичен потребителски интерфейс) -библиотеки като Tk, както и много други.

Тъй като Python е език, който се интерпретира, се спестява значително време за разработка, тъй като не са необходими компилиране и свързване (*linking*) за тестването на дадено приложение. Освен това, бидейки интерпретируем език с идеология сходна с тази на Java, приложение, написано на него, е сравнително лесно преносимо на множеството от останали платформи (или операционни системи).

Програмите, написани на Python, са доста компактни и четими, като често те са и по-кратки от еквивалентните им, написани на C/C++. Това е така, тъй като:

- наличните сложни типове данни позволяват изразяването на сложни действия с един-единствен оператор;
- групирането на изразите се извършва чрез отстъп, вместо чрез начални и крайни скоби или някакви други ключови думи (друг език, използващ такъв начин на подредба, е Haskell);
- не са необходими декларации на променливи или аргументи.
- Python съдържа прости конструкции, характерни за функционалния стил на програмиране, които му придават допълнителна гъвкавост

Всеки модул на Python се компилира преди изпълнение до код за съответната виртуална машина. Този код се записва за повторна употреба като ***.pyc*** файл.

Програмите написани на Python представляват съвкупност от файлове с изходен код. При първото си изпълнение този код се компилира до байткод, а при всяко следващо се използва кеширана версия. Байткодът се изпълнява от интерпретатор на Python.

- Строго типизиран (strong typing) – При несъответствие между типовете е необходимо изрично конвертиране.
- Динамично типизиран (dynamic typing) – Типовете на данните се определят по време на изпълнението. Работи на принципа duck typing – Оценява типа на обектите според техните свойства.
- Използва garbage collector – вътрешната реализация на езика се грижи за управлението на паметта.
- Блоковете се формират посредством отстъп. Като разграничител между програмните фрагменти използва нов ред.

Версии

Езикът се обновява често. В момента се тече преход от версия 2 към 3. Версия 3 съдържа много промени и поради това е несъвместима с по-старите реализации на езика. Повечето налични библиотеки и програми са написани и работят за версия 2.

Примерна програма

Това е сорс кодът на една примерна програма на python, която разпечатва на екрана *Hello Python!*

```
#!/usr/bin/python

print('Hello Python!')
```

Аритметични операции

Python притежава обичайните аритметични оператори (+ , - , * , / , %), като ** е допълнителен оператор, който се използва за степенуване, например 5**3 == 125.

Поведението на операцията за деление се променя значително с течение на времето[4]:

- Python 2.1 и по-ранни версии заимстват делението от езика C. Операторът / изпълнява целочислено деление, ако и двете операнди са цели числа, и деление с плаваща запетая в противния случай. Целочисленото деление закръглява към нулата, например 7 / 3 == 2 и -7 / 3 == -2.
- Python 2.2 променя целочисленото деление, така че да осъществява закръгляването надолу (към минус безкрайност), например 7 / 3 == 2 и -7 / 3 == -3. В тази версия е въведен операторът //, който извършва деление със закръгляване надолу (към по-ниската целочислена стойност). Неговото поведение може да се подразбере от следните примери: 7 // 3 == 2, -7 // 3 == -3, 7.5 // 3 == 2.0 и -7.5 // 3 == -3.0.
- Python 3.0 променя оператора / , така че той винаги да изпълнява ролята на деление с плаваща запетая. Обобщено, във версиите преди Python 3.0 / представлява „класическо деление“, докато във версия 3.0 / е деление на реални числа, а // е деление със закръгляване надолу (към по-ниската целочислена стойност).

Python предлага round функция за закръгляване на десетични към цели числа. Версиите преди Python 3.0 използват закръгляване далеч от нула: round(0.5) би дало като резултат 1.0 (закръгляване нагоре), съответно round(-0.5) връща −1.0 (закръгляване надолу)[6]. Python 3.0 използва закръгляване към най-близко четно число: round(1.5) връща като резултат 2, съответно round(2.5) отново е равно на 2[7].

Python позволява използването на булеви изрази с множество отношения на равенство, по начин, който е в съответствие със стандартната употреба на равенства в математиката. Например, изразът a < b < c проверява дали a е по-малко от b, и дали b е по-малко от c. Езиците, произхождащи от C, интерпретират този израз по различен начин: първо се пресмята a < b, връщащо като резултат 0 или 1, след което тази стойност се сравнява със c.

Поради своята богата математическа библиотека, Python често е използван като скриптов език с научна насоченост, с цел да помага при проблеми от сорта на обработка и управление на числови данни.

Преглед на функциите

- За разлика от много други езици Python не изисква декларации на променливи, те се създават при инициализацията им.
- Python разполага с вградени в самия език структури като: комплект (tuple), списък (list) и речник **(dictionary/map). Типът на променливата се определя от типа на присвоените стойности, в това отношение прилича на BASIC и се различава от много други програмни езици. Присвояването на различни стойности е коректно и това води до промяна на типа на променливата спрямо последната и присвоена стойност.

Комплект (tuple)

Комплекта се състои от елементи, които могат да бъдат произволни: прости константи или обекти. Елементите на комплекта не е задължително да са еднородни.

Комплектите (подобно на символните низове) са непроменливи: не е възможно да се присвои стойност на даден елемент от комплекта. Възможно е създаването на комплект, който съдържа в себе си променливи обекти – например списък (list).

Комплектите могат да бъдат вложени един в друг. Дефинират се чрез изброяване със запетайка, поставени в кръгли скоби.

(1, 4, 1, 78)
(3.14, ‘ string ’, -5)

Всички елементи са индексирани от нулата. За получаването на i-тия елемент трябва да се посочи името на комплекта и след това индексът i в квадратни скоби.

Пример:
t = (0, 1, 2, 3, 4)
print t[0], t[-1], t[-3]
Резултат: 0 4 2

По този начин комплектът може да се нарече вектор – константа, ако елементите му са били еднородни.

Списък (list)

Списъците са съставен (compound) тип данни, групиращ стойности (елементи), които сами по себе могат да бъдат от различен тип и могат да бъдат подменени след тяхната декларация. Възможно е присвояване на изрезки, като това може да промени и размера на списъка. Може да се вмъква друг списък в настоящия, да се добавят единични елементи, премахват и т.н.

Дефинира се чрез изброяване на елементите със запетая, поставени между квадратни скоби.
Пример:
[2, 4.87, ‘s’] [‘string’, (2,4,1), [5,8]]
Достъп до елементите се осъществява както и в комплекта.
Пример:
list = [1, ‘w’, (2,8), [0,3,4]] print list[0], list[1], list[-2], list[-1][0]
Резултат: 1 w (2,8) 0

Речник (dictionary/map)

Речникът е неподредено множество от двойки (ключ, стойност), като ключа е константа от всеки тип (най-вече символни низове). Комплекти могат да бъдат използвани за ключове само в случай, че не използват променливи обекти. Не може да се използва списък (list), тъй като той притежава възможността да се променя.

Основните операции са:

- Съхраняване на стойност под някакъв ключ (съхраняването на две стойности под един и същи ключ е невъзможно, запазва се втората такава подадена със съответния ключ).
- Извличане на стойността при подаване на ключа.
- Премахване на двойка ключ – стойност чрез „del“.
- Методът keys() на обект от тип речник връща списък с всички използвани ключове, а has_key() проверява дали даден ключ е в речника.

Чифт големи скоби „{ }“ създават празен речник.
Пример:
d = {'a': 1, 'b': 3, 5: 3.14, 'name': 'John'}
d['b'] = d[5]
print d['a'], d['b'], d[5], d['name'] Резултат: 1 3.14 3.14 John
За да добавите нова двойка „ключ“:"стойност", е достатъчно да присвоите на елемент с нов ключ съответната стойност:
d['new'] = 'new value'
print d
Резултат: {'a':1, 'b':3, 5:3.14, 'name':'John', 'new':'new value'}

- Python не поддържа указатели, динамична памет и адресна аритметика (подобно на Java).
- Присвояване от една променлива към друга (Пример: a = 2; b = a;) има своите особености в Python. В случаи на присвояване на обекти от тип числа или низове се създава нов обект (под „нов обект“ да се разбира памет за съхранение на стойност от всеки тип). Копираното съдържание се присвоява от променливата 'a' в променливата 'b'(семантика на копирането – променливите от двете страни на оператора значат два независими обекта).

Ако от дясната страна стои стойност от тип клас, списък, речник или комплект присвояваме само връзката (указателя) към адреса, където е стойността на обекта, двата обекта сочат към една и съща памет – при промяна на стойността на едната се променя и другата (семантика на указатели).

Пример:
a = 5; b = a; b = 3
print ' семантика на копирането: a=', a, 'b=', b
a = [2,5]; b = a; b[0] = 7
print ' семантика на указателите: a=', a, 'b=', b
Резултат:
семантика на копирането: a = 5 b = 3
семантика на указателите: a= [7,5] b= [7,5]

Управляващи конструкции

If-конструкция

Пример:
>>> # x = int(raw_input("Моля, въведете едно число: "))
>>> if x < 0:
... x = 0
... print ’Отрицателно, сменено на нула’
... elif x == 0:
... print ’Нула’
... elif x == 1:
... print ’Едно’
... else:
... print ’Повече’
...
if <условие1>: <оператор1>
[elif <условие2>: <оператор2>]*
[else: <оператор3>]

При истина на <условие1> , ще се изпълни <оператор1> и ще се игнорират клоновете elif и else. В противен случай, ако е вярно <условие2>, ще се изпълни <оператор2>, а клонът else ще се игнорира.

Иначе ще се изпълни <operator3>.

Ключовата дума elif е съкращение от else-if, може да има няколко elif части, else не е задължителна. Поредица от if – elif може да замени switch - case конструкцията, която откриваме и в други езици.

For цикъл

В Python операторът for итерира върху елементите на каквато и да е редица (напр. символен низ или списък) по реда на елементите в редицата.

Пример:
...list = ['чета', 'статия', 'в УикипедиЯ']
>>> for element in list:
... print element, len(element)
...
чета 4

статия 6
в УикипедиЯ 11

Не е безопасно да се променя редица, върху която се итерира в цикъл. Ако трябва да се променя редица, върху която се итерира (например списък), трябва това да се итерира върху негово копие. Конвенцията за изрязване улеснява това частично.

Пример:
```
>>> for element in list[:]: #Прави копие чрез изрязване
... if len(element) > 6: list.insert(0, element)
...
>>> list
['в УикипедиЯ', 'чета', 'статия', 'в УикипедиЯ']
```

While цикъл

while <условие>:
<оператор1>
[else: <оператор2>]

Цикъл „докато“: <Оператор1> ще се изпълнява, докато е истина <условие>. При нормално завършен цикъл, т.е. без прилагане на break, ще се изпълни <оператор2>.

Пример: Числа на Фибоначи
```
... a, b = 0, 1
>>> while b < 10:
... print b
... a, b = b, a+b
...
1
1
2
3
5
8
```

Дефиниране на функции

Ключовата дума *def* въвежда дефиниция на функция. Тя трябва да бъде последвана от име и списък от параметри, поставени в скоби. Тялото на функцията започва на следващия ред и трябва да бъде с отстъп. Първият ред оператори може да бъде символен низ (*документационен символен низ* – docstring), който представлява документацията (описанието) на функцията. Можем да създадем функция, която изписва числата на Фибоначи в произволна граница:

```
>>> def fib(n): # изписва числата на Фибоначи до n
... „Изписва числата на Фибоначи до n“
... a, b = 0, 1
... while b < n:
... print b,
```

```
... a, b = b, a+b
...
>>> # Сега да извикаме току-що дефинираната функция:
... fib(2000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

Източници

1. Venners, Bill (13 January 2003). "The Making of Python". *Artima Developer*. Artima. Retrieved 22 March 2007.
2. van Rossum, Guido (20 January 2009). "A Brief Timeline of Python". *The History of Python*. Google. Retrieved 20 January 2009.
3. "Why was Python created in the first place?". *General Python FAQ*. Python Software Foundation. Retrieved 22 March 2007.
4. Zadka, Moshe; van Rossum, Guido (11 March 2001). "PEP 238 – Changing the Division Operator". *Python Enhancement Proposals*. Python Software Foundation. Retrieved 23 October 2013.

Препратки

- Официалният сайт на Python
- Български превод на ръководството за Python
- A Byte of Python – свободна книга за Python
- Ръководство по Питон 2.0.1 Превод на „Python Tutorial“ на български език от Калоян Доганов