به نام خدا

جزوه

تهیه‌کننده: س. موسوی

mousavi1954@gmail.com

۱۹ آذر ۱۴۰۳

پیش‌گفتار

تاریخچه زبان پایتون

اگر بخواهیم نگاهی به سیر تاریخی شکل‌گیری زبان برنامه نویسی Python بیاندازیم شاید باید به سال ۱۹۸۲ بازگردیم، زمانی که Guido Van Rossum خالق زبان برنامه‌نویسی پایتون فعالیت خود را در مؤسسه‌ی تحقیقاتی مرکز ریاضیات و علوم کامپیوتری CWI در آمستردام هلند آغاز کرد و در سال ۱۹۹۱ اولین نسخه زبان پایتون ارائه داد.

داستان نامگذاری:

ون روسوم پیش از هر چیز، تلاش کرد نامی مناسب برای زبان جدیدی که در صدد طراحی آن بود پیدا کند و با توجه به این که این زبان جدید از دل پروژه‌ی ABC بیرون می‌آمد، در ابتدا قصد داشت آن را B بنامد، اما متوجه شد زبانی به همین نام وجود دارد. پس از آن که روسوم بسیاری از پیشنهادات اعضای گروه را در مورد نام زبان جدید رد کرد تصمیم گرفت اولین نامی را که به ذهنش رسید انتخاب کند، که به طور اتفاقی به یاد کمدی محبوبش که آن روزها از شبکه‌ی BBC با نام Monty Python's Flying Circus پخش می‌شد افتاد و به این ترتیب نام پایتون را برای پروژه‌ی جدید خود انتخاب کرد.

تا مدت‌ها روسوم اجازه نمی‌داد که از تصویر پایتون که گونه‌ای مار است به عنوان نماد این زبان استفاده شود و اولین بار انتشارات O'Reilly که همیشه تصویر یک جانور را روی کتاب‌های خود قرار می‌دهد، از تصویر یک مار روی کتاب آموزش برنامه‌نویسی به زبان پایتون استفاده کرد و بعدها نیز اغلب از تصویر یک مار به عنوان نماد پایتون استفاده شد.

نسخه‌های زبان پایتون

- در حال حاضر پایتون دو نسخه دارد. نسخه 2.x و نسخه 3.x

- پشتیبانی از دو نسخه‌ی 2 و 3 به صورت موازی در کنار هم ادامه دارد، با این حال بر اساس قراردادهای صورت گرفته توسعه نسخه‌ی 2 تنها تا شماره‌ی 2.7 ادامه می‌یابد و پشتیبانی از آن فقط تا سال ۲۰۲۰ ادامه داشت.

- قاعدتاً پس از انتشار نسخه‌ی 3 پایتون تمام برنامه‌ها و کتابخانه‌های نسخه‌ی قبلی باید به نسخه‌ی جدید ارتقا می‌یافتند، با این حال اِعمال قابلیت‌ها و گرامر جدید در نسخه‌های قبلی و مهاجرت شرکت‌های بزرگ به نسخه‌ی جدید بسیار زمان بر بود. از طرفی نسخه‌ی جدید هم قابلیت Backward Compatibility یا سازگاری با نسخه‌های پیشین را نداشت و در صورتی که ایرادی در نسخه‌های 2 وجود داشت، کاربران نمی‌توانستند با استفاده از نسخه‌ی جدید بر آن ایرادات فائق آیند. بر همین اساس تیم توسعه‌ی زبان برنامه‌نویسی پایتون تصمیم گرفت در یک دوره‌ی زمانی محدود توسعه‌ی نسخه‌ی 2 را ادامه دهد.

چرا زبان برنامه‌نویسی پایتون؟

زبان برنامه‌نویسی پایتون به دلایل زیر انتخاب شده است.

۱. یادگیری آن نسبتاً ساده است، چرا که گرامر آن به زبان محاوره (انگلیسی) نزدیک است.

۲. دارای کاربردهای متنوع است.

۳. متن باز و رایگان است.

۴. دارای کتابخانه‌های وسیع و متنوعی است.

۵. روی همه‌ی سیستم عامل‌ها نظیر Windows و Mac.Os و Linux کار می‌کند.

۶. کُدهای آن از یک سیستم به سیستم دیگر قابل حمل است.

۷. خاصیت شی‌گرایی[1] دارد.

۸. رشد سریع و استفاده آن در دنیا

تمام کُدهای این جزوه در محیط JupyterLab نوشته و اجرا شده است. و درج آنها در جزوه هم به همان صورت محیط یاد شده حروف‌نگاری شده است. بنابراین اجرای پاره‌ای از سلول[2]ها نیازمند اجرای سلول[های] پیشین است.

در خاتمه بر بنده فرض است که از آقای وفا خلیقی بخاطر فراهم نمودن بسته‌ی زی‌پرشین[3] که امکان نوشتن متون فارسی در محیط لاتک[4] را ایجاد نموده است، صمیمانه سپاسگزاری کنم.

1. Object Oriented
2. Cell
3. XePersian
4. LaTeX

مسلماً نوشته حاضر خالی از خلل نیست، نویسنده از هر گونه اظهار نظر و پیشنهادی برای اصلاح و ارتقاء آن استقبال و استفاده خواهد نمود.

فهرست مطالب

پیش‌گفتار ج

۱ کلیات زبان ۱

۱-۱ مقدمه ۱

۱-۱-۱ محاسبات اولیه ریاضی ۱

۱-۱-۲ انواع داده‌ها یا مقادیر ۲

۲ محیط‌های کنترلی ۴

۲-۱ مقدمه ۴

۲-۲ شرط ۴

۲-۲-۱ شرط‌های تو در تو ۶

۲-۲-۲ دستور شرط‌های مفصل ۷

فهرست شکل‌ها

۲-۱ روندنمای بلوک if-else ۵
۲-۲ روندنمای شرط تو در تو if-else ۶
۲-۳ روندنمای شرط elif ۷

فهرست جدول‌ها

۱ کلیات زبان

۱-۱ مقدمه

در این فصل موارد اولیه زبان آورده می‌شود. که عبارتند از:

- محاسبات اولیه ریاضی
- انواع داده‌ها یا مقادیر
- معرفی متغیر، عبارت
- معرفی عملگرها

۱-۱-۱ محاسبات اولیه ریاضی

در پایتون می‌توان چهار عمل اصلی و توان را مثل یک ماشین حساب محاسبه نمود. برای جمع از علامت (+)، برای تفریق از علامت (-)، برای عمل ضرب از علامت (*) و برای تقسیم از علامت (/) استفاده می‌شود. به چند مثال زیر توجه کنید.

```
In [1]:  4+5
         5-7
         3*5
         5/4
         9/3
```

```
Out[1]:  9
         -2
         15
         1.25
         3.0
```

توجه: حاصل تقسیم در همه حالت‌ها، عدد اعشاری است.
برای نشان دادن نتیجه اجرای کُدها روی صفحهٔ نمایش می‌توان از تابع print() استفاده نمود.
در مورد تقسیم دو عملگر (//) و (%) هم وجود دارد. اولی قسمت صحیح عمل تقسیم و دومی باقیمانده تقسیم را بدست می‌دهد. مثال:

```
In [1]:  print(17/5)
         print(17//5)
         print(17%5)
```

```
Out[1]:  3.4
         3
         2
```

و اگر مقدار ۱۷/۵− را قرار دهیم، نتیجه می‌شود.

```
In [2]:  print(-17/5)
         print(-17//5)
         print(-17%5)
```

```
Out[2]:  -3.4
         -4
         3
```

برای علامت توان از (**) استفاده می‌شود.

```
In [3]:  5**3
```

```
Out[3]:  125
```

۱-۱-۲ انواع داده‌ها یا مقادیر

در پایتون دو دسته داده داریم. داده‌های عددی و داده‌های غیر عددی. داده‌های عددی سه نوع[1] هستند.

- صحیح[2]
- اعشاری[3] یا دارای نقطه اعشار

1. type
2. integer
3. float

- مختلط[1]

و داده‌های غیر عددی

- رشته[2]
- بولی[3]

در مورد داده‌های عددی، به عنوان مثال ۲۳ یک عدد صحیح و ۷/۷ یک عدد اعشاری است. عدد .۵ دارای نقطه اعشار است، بنابراین عدد اعشاری محسوب می‌شود.
فرم کلی عدد مختلط به صورت $x + y$j است. قسمت موهومی عدد همان مؤلفه‌ای است حرف j را دارد.
در پایتون تابعی وجود دارد که نوع داده‌ها را مشخص می‌کند. این تابع به صورت type() است. اکنون به چند مثال توجه کنید.

```
In [1]:  print(type(5))

Out[1]:  <class 'int'>

In [2]:  print(type(7.7))

Out[2]:  <class 'float'>

In [3]:  print(type(2+3j))

Out[3]:  <class 'complex'>
```

1. complex
2. string
3. boolean

۲

محیط‌های کنترلی

۲-۱ مقدمه

در یک مجموعه کُد از پایتون، قواعد اجرای آن به شرح زیر است.

۱. دستورات از بالا به پایین اجرا می‌شود، تا برنامه به پایان برسد.

۲. هر دستور فقط یکبار اجرا می‌شود.

۳. اجرا به دستورات قبل از خود باز نمی‌گردد.

در پاره‌ای اوقات لازم است که بر اساس تصمیمی یک و یا چند دستور اجرا نگردد و از روی آنها عبور کنیم.
در برخی از زمان‌ها لازم است که یک و یا چند دستور بیش از یکبار اجرا شود. به عبارت دیگر لازم است که به دستورات قبلی مراجعه گردد.
بنابراین ساز و کارهایی لازم است که پیش‌فرض‌های بالا را تغییر دهد. در زبان‌های برنامه‌نویسی دو مکانیزم وجود دارد که خواسته ما را تامین می‌کنند.
اگر بخواهیم تحت شرایطی یک یا چند سطر اجرا نشود از محیط شرط استفاده می‌شود و اگر قرار باشد که یک یا چند سطر بیش از یک بار اجرا شود از محیط حلقه استفاده می‌کنیم.

۲-۲ شرط

به روندنمای زیر توجه کنید گرامر و یا syntax شرط به‌صورت زیر است. به‌عنوان مثال:

In [1]:
```
# python program to illustrate If statement
i = 10
if (i>15):
    print("10 is less than 15")
print("I am Not in if")
```

Out[1]:
```
I am Not in if
```

ساختار شرط به شرح زیر است.

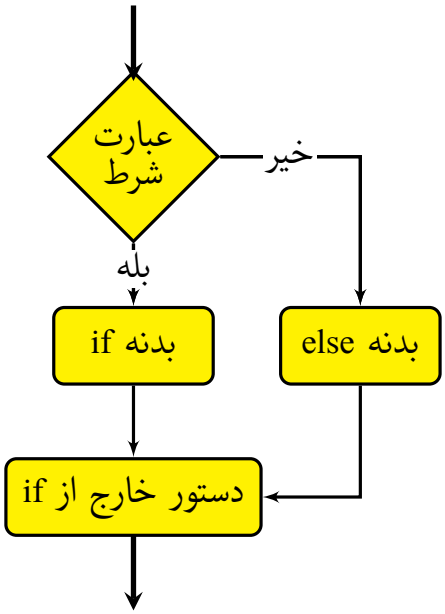
۱. استفاده از کلید واژه if

۲. استفاده از یک عبارت boolean مثل $x > y$ که نتیجه آن درست یا غلط است و بر اساس آن تصمیم‌گیری صورت می‌گیرد.

۳. استفاده از نماد «:»[1] ضروری است وگرنه برنامه خطا می‌دهد.

۴. بدنه شرط: یعنی دستوراتی که داخل شرط قرار می‌گیرند و کنترل روی آنها اعمال می‌شود. دستوراتی که در بدنه شرط قرار دارند، دارای تورفتگی[2] هستند.

توجه: علامت (#) برای ارایه توضیحات به کار می‌رود که اجرای برنامه آن را نادیده می‌گیرد.
می‌توان شرط را بسط بیشتری داد. بدین معنی که اگر عبارت شرطی درست نبود، کنترل برنامه به کجا منتقل می‌شود؟ برای مدیریت این کار از else استفاده می‌شود. روندنمای آن در نمودار ۲-۳ آمده است.



شکل ۲-۱: روندنمای بلوک if-else

1. colon
2. indentation

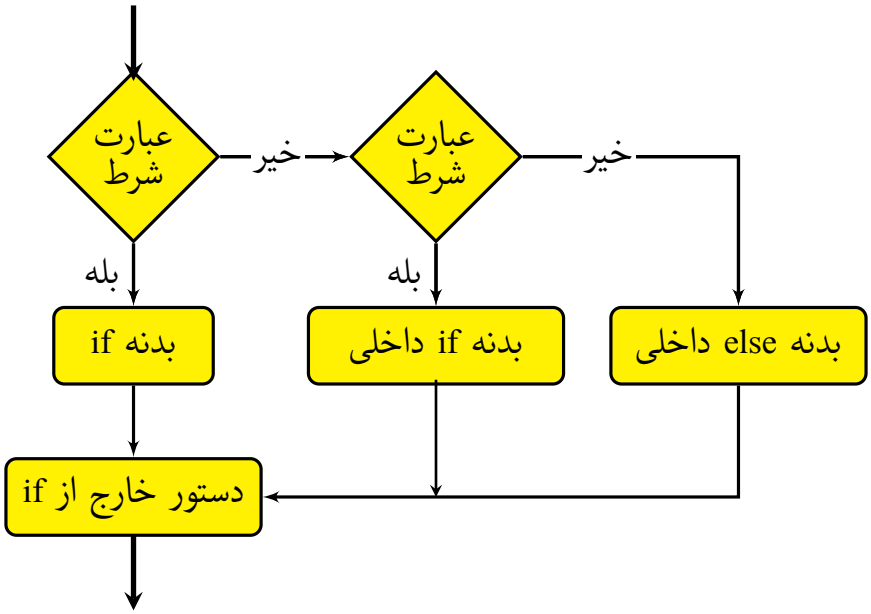
مثال:

In [1]:
```
number=9
if number%2==0:
    print("The number is even.")
else:
    print("The number is odd.")
```

Out[1]:
```
The number is odd.
```

۲-۲-۱ شرط‌های تو در تو

می‌توان در داخل بدنه یک شرط دستورات مختلفی را داشت. از جمله یک شرط می‌تواند درون بدنه یک شرط دیگر واقع شود. به این شرط‌ها اصطلاحا شرط‌های تو در تو[1] گویند.



شکل ۲-۲: روندنمای شرط تو در تو if-else

مثال:

In [2]:
```
i, j, k = 7, 5, 3
if i > k:
    if j > k:
        print('i and j are greater than k')
else:
    print('i is less than or equal to k')
```

Out[2]:
```
i and j are greater than k
```

مثال دیگر:

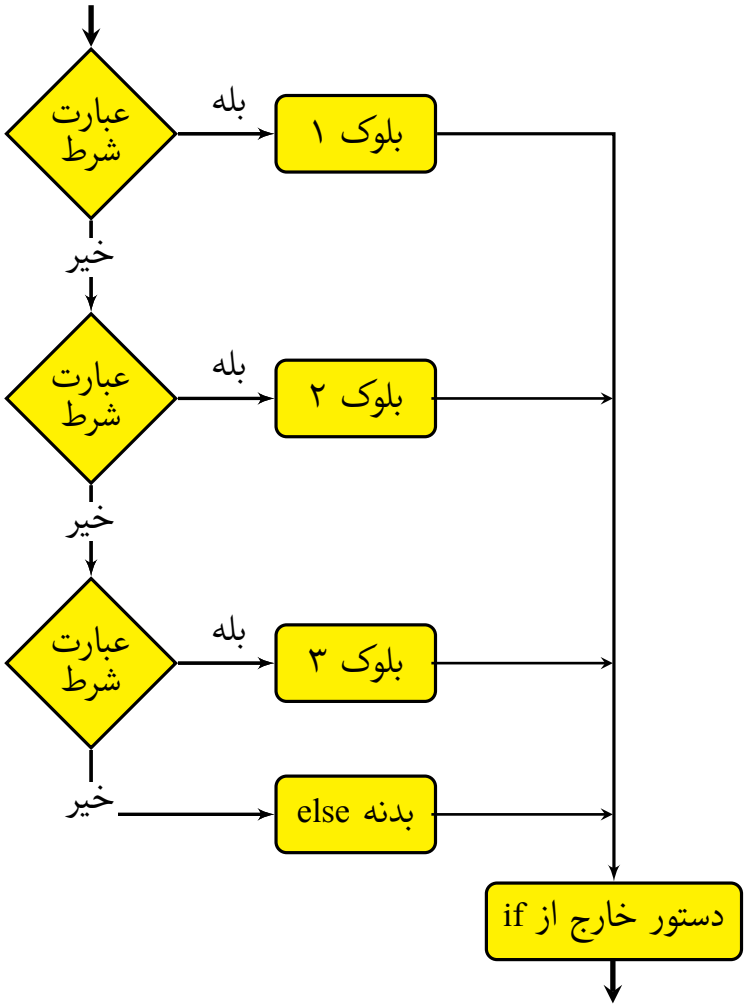
1. nested conditions

In [1]:
```
# if..else chain statement
letter = "A"
if letter == "B":
    print("letter is B")
else:
    if letter == "C":
        print("letter is C")
    else:
        if letter == "A":
            print("letter is A")
        else:
            print("letter isn't A, B and C")
```

Out[1]:
```
letter is A
```

۲-۲-۲ دستور شرط‌های مفصل

در شرط‌هایی که تعداد تصمیم‌ها نسبتا زیاد است می‌توان از دستور elif استفاده نمود.



شکل ۲-۳: روندنمای شرط elif

مثال:

In [1]:
```
a = 7
b = 9
c = 3
if((a>b and a>c) and (a != b and a != c)):
    print(a, " is the largest")
elif((b>a and b>c) and (b != a and b != c)):
    print(b, " is the largest")
elif((c>a and c>b) and (c != a and c != b)):
    print(c, " is the largest")
else:
    print("entered numbers are equal")
```

Out[1]:
```
9  is the largest
```