

مبانی برنامه‌ریزی کامپیوتر (مبانی سرویسی) Fundamentals of Programming (Python)



سمانه رستگاری
جلسه 2: مقدمات برنامه‌سازی

عناوین

مقدمه

مقادیر متغیرها و انواع داده‌ها

عملگرها و تقدم عملیات

تبدیل گونه‌ها

دستورات ورودی و خروجی

آشنایی با کدنویسی خوانا

(فصل‌های 1 و 2 از کتاب اصلی)

مقدمه

- آنچه در یک الگوریتم اهمیت دارد زبان دقیق آن است که نتوان از آن برداشت‌های متفاوت داشت و یا سؤال برانگیز باشد پس باید غیرمبهم باشد برخلاف زبان طبیعی.
- سه ویژگی دیگر یک الگوریتم، جزئیات کافی، ترتیب مراحل و خاتمه‌پذیر بودن عملیات است.
- خوب بودن یک الگوریتم بستگی به تعداد مراحل آن ندارد بلکه سادگی تعمیم و فهم آن مهم است.
- الگوریتمی بنویسید که سه مقدار عددی را از ورودی دریافت کرده و میانگین سه عدد را محاسبه کند.

- 0- شروع کن
- 1- A، B و C را بگیر
- 2- $S \leftarrow A+B+C$
- 3- $M \leftarrow \frac{S}{3}$
- 4- M را تا دو رقم اعشار بنویس.
- 5- پایان.

تمرین

- ◎ الگوریتمی بنویسید که شعاع دایره را دریافت کرده، محیط و مساحت آنرا حساب کند.
- ◎ الگوریتمی بنویسید که دومقدار را از ورودی دریافت کرده، مقدار بزرگتر را چاپ کند.

مقدمه (ادامه)

◎ یک برنامه، دنباله ای از دستورات است که چگونگی انجام محاسبات را مشخص می کند؛ محاسبات می توانند از نوع ریاضی مانند حل دستگاه معادلات یا محاسبات نمادین از قبیل یافتن و جایگزین کردن کلمه در یک متن باشند.

◎ دستورات پایه ای در تمام زبان ها:

○ Input: گرفتن داده ها از صفحه کلید، فایل یا دیگر واحدهای ورود اطلاعات.

○ Output: نمایش داده ها بر روی صفحه نمایش یا ارسال آنها به یک فایل یا دیگر واحدهای خروج اطلاعات.

○ Math: انجام دادن اعمال ریاضی بنیادی مانند ضرب و جمع.

○ Conditional: بررسی شروط خاص و اجرای دنباله ای از دستورات بر اساس آن شرایط.

○ Repetition: انجام برخی اعمال در چندین بار در عمل مورد تکرار.

◎ تقریباً تمام برنامه ها با کمک همین دستورات

برنامه نویسی،
شکستن کار بزرگ
پیچیده به کارهای
کوچکتر است

مقدمه (ادامه)

خطاهای برنامه‌نویسی را bug می‌نامند و به عمل جداسازی و تصحیح آن خطاها debugging گفته میشود. ◎

انواع خطا: ◎

◎ Syntax errors (نحوی یا گرامری): پایتون تنها برنامه‌هایی را که از نظر گرامر درست هستند اجرا می‌کند.

◎ Runtime errors (زمان اجرا): تا زمان اجرا مشخص نمی‌شوند. این خطاها را اعتراض هم می‌نامند زیرا آنها معمولاً نشان می‌دهند که اتفاق اعتراض آمیز و بدی رخ داده است.

◎ Semantic errors (معنایی): کامپیوتر هیچ خطایی گزارش نمی‌کند ولی هدف برنامه صورت نمی‌پذیرد؛ هدف شما چیزی غیر از آنچه در برنامه نوشته‌اید است و معنای برنامه اشتباه است.

◎ فرایند debugging (اشکال زدایی) همانند یک علم تجربی است. به محض اینکه شما ایده‌ای در مورد اشکال کار به دست می‌آورید، برنامه را تصحیح کرده و دوباره تلاش میکنید. اگر فرضیه شما درست باشد، آنگاه می‌توانید نتیجه تغییر و تحول را پیش‌گویی کنید و یک قدم به برنامه قابل اجرا و صحیح نزدیک تر شوید، اما اگر فرضیه شما غلط باشد مجبورید ایده جدیدی ارائه دهید

مقدمه (ادامه)

◎ انواع قوانین نحوی:

1. مربوط به ساختار عبارات (نحوه ی آرایش توکن ها)
2. مربوط به نشانه‌ها (token)

◎ نشانه یا توکن کوچکترین واحد هر زبان است مثل کلمه در زبان طبیعی و اعداد در زبان ریاضی.

$3 + 4 = 7$ syntax ✓

$3 += 6$ syntax error (ساختاری)

$3 + 4 \$ 7$ syntax error (نشانه‌ای)

مقادیر

یک مقدار (value) یکی از موارد پایه‌ای است که برنامه با آن کار میکند مثل یک رشته یا یک عدد:

String

"Hello, World!" یا 2

Integer

مقادیر به دسته‌های مختلفی طبقه‌بندی می‌شوند: انواع داده‌ای (data types)

اگر مطمئن نیستید که یک مقدار از چه دسته‌ای است از تابع type استفاده کنید.

```
>>> type("Hello, World!")  
<type 'string'>  
>>> type(17)  
<type 'int'>
```

اعداد اعشاری به نوعی تحت عنوان float تعلق دارند. ولی مقادیری شبیه به "3.2" یا "17" از نوع رشته!

```
>>> type("17")  
<type 'str'>
```

رشته‌ها به راحتی قابل تشخیص هستند زیرا آنها در بین دو علامت جفت کوتیشن (") و یا کوتیشن (') قرار می‌گیرند.

در نوشتن اعداد صحیح بزرگ از کاما استفاده نکنید چون معنی دیگری پیدا می‌کند.

متغیرها

متغیر (variable) نامی است که به یک مقدار اشاره می کند. ◎

دستور نسبت دهی یک متغیر جدید می سازد و مقداری را به آن نسبت می دهد. نشانه دستور انتساب (assignment) این علامت است: = ◎

```
>>> message = "What's up, Doc?"
```

```
>>> n = 17
```

```
>>> pi = 3.14159
```

نباید با دستور مقایسه تساوی (==) اشتباه گرفته شود. ◎

سوال: 1) چگونه مقدار متغیر را ببینیم؟ 2) چگونه نوع متغیر را چک کنیم؟ ◎

نوع یک متغیر، نوع مقداری است که متغیر به آن اشاره می کند. ◎

ما از متغیرها در برنامه برای به خاطر داشتن (remember) چیزها استفاده می کنیم ولی در طول زمان تغییر می کنند؛ یعنی مقدار متغیر می تواند تغییر کند و حتی نوع آن!! ◎

کلمات کلیدی و اسامی متغیرها

- نام متغیر باید با حرف یا _ (underscore) شروع شود و شامل حرف یا عدد باشد.
- اگرچه می‌تواند با حروف بزرگ شروع شود ولی طبق قرارداد، متغیرها با حروف کوچک شروع می‌شوند.
- چون underscore در ابتدای برخی اسامی معنی ویژه‌ای دارد بهتر است مبتدیان نام تمام متغیرها را با حروف کوچک شروع کنند.
- کلمات کلیدی، ساختار و قوانین زبان را تعریف می‌کنند و نمی‌توانند به عنوان اسامی متغیرها استفاده شوند. (class, if, else, while, ...)
- اگر نام متغیر درست و با معنی انتخاب شود به خواننده برنامه در درک بهتر آن کمک می‌کند.

دستورات

◎ یک دستور (statement) عبارتی است که مفسر پایتون قادر است آن را اجرا کند. تا به حال ما دو نوع از دستورات را دیده ایم: چاپ و نسبت دهی.


◎ وقتی دستوری را در خط فرمان تایپ می کنید، پایتون آن را اجرا می کند و اگر نتیجه ای وجود داشته باشد آن را نمایش می دهد. نتیجه دستور چاپ یک مقدار است اما دستورات نسبت دهی نتیجه ای تولید نمی کنند.

◎ سوال: خروجی اسکرپت زیر چیست؟

```
print 1  
x = 2  
print x
```

◎ دستورات for، if، import، while و ... را در آینده نزدیک خواهیم دید.

ارزیابی عبارت‌ها

یک عبارت ترکیبی از مقادیر، متغیرها و عملگرها است. اگر شما عبارتی را در خط فرمان تایپ کنید، مفسر آن را ارزیابی می‌کند و نتیجه را نمایش می‌دهد. 

```
>>> 1 + 2
```

```
3
```

```
>>> pi = 3.14
```

```
>>> pi
```

```
3.14
```

```
>>> type(pi)
```

```
<class 'float'>
```

عملگرها و عملوندها

◎ عملگرها (operators)، نمادهای ویژه‌ای هستند که محاسباتی مثل جمع، تفریق، ضرب، تقسیم و توان را نشان می‌دهند. مقادیری که عملگرها استفاده می‌کنند عملوند (operand) نامیده می‌شوند.

`20+32` `hour-1` `hour*60+minute` `minute/60` `5**2` `(5+9)*(15-7)`

◎ در پایتون نسخه سوم به بعد عملگر / تقسیم اعشاری است و اگر بخواهید فقط خارج قسمت را بگیرید از عملگر // استفاده کنید که همیشه به کمتر گرد می‌کند.

`>>> 6/4`

`1.5`

`>>> 6//4`

`1`

◎ برای گرفتن باقیمانده تقسیم از عملگر % استفاده می‌شود.

`>>> 6%4`

`2`

تبدیل نوع داده


- ◎ پایتون مجموعه ای از توابع پیش ساخته را فراهم می کند که می توانند مقادیر را از یک نوع به نوع دیگر تبدیل کنند.
- ◎ تابع `int()` هر مقداری را می گیرد و در صورت امکان به یک عدد صحیح تبدیل می کند و در غیر این صورت پیغام خطایی را نمایش می دهد.
- ◎ تابع `float()` تابعی است که اعداد صحیح و رشته عددی را به اعداد اعشاری تبدیل می کند.
- ◎ تابع `str()` تابعی است که اعداد صحیح و اعداد اعشاری را به نوع رشته تبدیل می کند.

ترتیب عملگرها


وقتی که بیشتر از یک عملگر در عبارتی استفاده شود، ترتیب ارزیابی آنها به قوانین اولویت بستگی دارد. پایتون از قوانین اولویتی مشابه عملگرهایی که در ریاضیات استفاده می شوند، پیروی می کند. (قانون PEMDAS)

1. Parentheses (پرانتزها)
 2. Exponentiation (توان)
 3. Multiplication and Division (ضرب و تقسیم اولویت یکسان دارند)
 4. Addition and Subtraction (جمع و تفریق اولویت یکسان دارند)
- عملگرهای با تقدم یکسان، از چپ به راست ارزیابی می شوند.

دستور ورودی

یک تابع پیش‌ساخته (built-in) در پایتون برای گرفتن ورودی از کاربر وجود دارد: `input` 

```
name = input("Please enter your name: ")
```

خروجی این تابع همیشه رشته است بنابراین اگر یک عدد را از کاربر می‌گیرید می‌توانید با توابع `int` و `float` که پیش‌تر گفته شد آن را تبدیل کنید. 

```
age = int(input("Please enter your age: "))
```


دستور خروجی

یک تابع پیش‌ساخته (built-in) در پایتون برای نمایش روی مانیتور وجود دارد: `print`

```
print("Your name is: ", name)
```

خروجی این تابع همیشه روی مانیتور نمایش داده می‌شود که ابتدا پارامتر اول تابع چاپ شده و بعد مقدار متغیر پارامتر بعدی نشان داده می‌شود که این متغیر باید قبلاً مقداردهی شده باشد.

حالا برنامه محاسبه مساحت دایره را با دریافت شعاع آن از کاربر بنویسید.

عملیات بر روی رشته ها

◎ عملگر + باعث الحاق رشته ها می شود.

```
fruit = "banana"  
bakedGood = " nut bread"  
print (fruit + bakedGood)
```

◎ خروجی این کد چیست؟

◎ عملگر * نیز بر روی رشته ها کار می کند و عمل تکرار را انجام می دهد.

```
print(3*'Fun')
```

◎ خروجی این کد چیست؟

آشنایی با کدنویسی خوانا

- توضیح (comment) در برنامه با # شروع می‌شوند. اطلاعاتی در برنامه که برای دیگر برنامه نویسان (و یا دیگر کسانی که کد مبداء برنامه را می‌خوانند) در نظر گرفته می‌شود و هیچ تأثیری در روند اجرای برنامه ندارند.
- هر برنامه حداقل شامل اسم برنامه نویس آن و کاری که قرار است انجام دهد می‌باشد؛ پس حداقل دو خط یادداشت دارد.
- بین عملگر و هر عملوند یک جای خالی (space) بگذارید.
- وقتی می‌خواهید بین عملگرهای یک تابع کاما بگذارید باید کاما به عملگر قبل از آن بچسبید و بعد از کاما یک جای خالی بگذارید.
- از tab برای فاصله‌های بیشتر از یک جای خالی منحصرأ استفاده کنید.
- خطوطی از کد که به لحاظ منطقی به هم مربوطند را پشت سر هم و با یک خط خالی از قسمت غیرمرتبط جدا کنید.