



پایتون

مقدماتی



فهرست:

- ۱.....دستور () input
- ۲.....استفاده از دستور input() در برنامه چه حُسنی دارد؟
- ۳.....تابع int()
- ۵.....یک اشتباه بسیار رایج
- ۶.....تابع float()
- ۷.....برتری دستور float() نسبت به int()
- ۷.....آشنایی با Comment
- ۷.....دلایل استفاده از Comment
- ۸.....نحوه معرفی Comment



دستور () input

با استفاده از دستور `input()` می‌توان در زمان اجرای برنامه، داده‌هایی را از طریق صفحه‌کلید از کاربر دریافت نمود و در ادامه اجرای برنامه از آنها استفاده کرد. به عبارت دیگر دستور `input()` این امکان را فراهم می‌نماید که برنامه در هر بار اجرا با مقادیر جدیدی اجرا شود که از کاربر دریافت می‌شوند.

نکاتی در رابطه با دستور `input()`

- برنامه در زمان اجرا با رسیدن به دستور `input()` منتظر می‌ماند تا کاربر داده‌ای را وارد نماید.
- با توجه به اینکه این دستور داده‌ای را از کاربر دریافت می‌نماید، به یک متغیر نیاز دارد تا داده دریافت شده را در آن نگهداری کند.
- این دستور مانند دستور `print()` می‌تواند یک عبارت متنی را قبل از دریافت داده از کاربر بعنوان راهنما به او نشان دهد تا کاربر در زمان اجرای برنامه بداند چه چیزی را باید به عنوان داده وارد کامپیوتر نماید. با توجه به اینکه این عبارت، یک عبارت متنی است، مسلماً باید داخل دو علامت گیومه یا نقل قول (") قرار گیرد.

توضیح دستور: `name = input("Enter your Name : ")`

- همانطور که در این خط از دستور مشاهده می‌کنید، برای دستور `input()` متغیری به نام `name` در نظر گرفته شده است. یعنی هر داده‌ای که کاربر در زمان اجرا وارد نماید، در این متغیر ذخیره خواهد شد.
- داخل پرانتز و در مقابل دستور `input` عبارت `Enter your Name:` نوشته شده است. این عبارت در زمان اجرای برنامه به کاربر نشان داده می‌شود تا کاربر بداند چه چیزی را باید به عنوان داده وارد کند. این عبارت به دلیل آنکه یک عبارت متنی است، در داخل دو علامت گیومه یا نقل قول (") قرار داده می‌شود.



استفاده از دستور `input()` در برنامه چه حُسنی دارد؟

برای روشن شدن حُسن استفاده از این دستور در برنامه، به مثال‌های مقابل توجّه نمایید.

```
10.py
File Edit Format Run Options Window Help

name = " Python "
print( "hi",name )

Ln: 6 Col: 0
```

- در برنامه اول کلمه **Python** در داخل برنامه به متغیر **name** اختصاص داده شده است. در نتیجه هر بار که این برنامه اجرا شود، عبارت **Hi Python** نمایش داده خواهد شد.

```
11.py
File Edit Format Run Options Window Help

name = input("Enter you Name : ")
print( "Hi ",name )

Ln: 5 Col: 0
```

- اما در برنامه دوم، مقدار متغیر **name** در هر بار اجرای برنامه از کاربر دریافت می‌شود. در نتیجه هر بار که برنامه اجرا می‌شود، عبارت جدیدی چاپ خواهد شد که به عبارت وارد شده توسط کاربر بستگی دارد.

نکته بسیار مهم: داده‌ای که توسط دستور `input()` از کاربر دریافت می‌شود حتی اگر یک عدد هم باشد، برای پایتون به عنوان یک داده از نوع **متنی** یا **String** شناخته می‌شود. به نمونه برنامه زیر توجّه کنید:

```
06.py
File Edit Format Run Options Window Help

a = input("Enter a number : ")
print( 3 * a )

Ln: 5 Col: 0
```

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help

Enter a number : 10
101010
>>>
```

سه بار عبارت 10 تکرار شده است

در این تمرین از کاربر خواسته شده تا یک عدد وارد نماید و کاربر هم عدد 10 را وارد کرده است. در ادامه، برنامه باید عدد دریافت شده از کاربر را در عدد 3 ضرب کرده و حاصل آن یعنی عدد 30 را چاپ نماید. اما در کمال تعجب مشاهده می‌شود که عبارت 101010 چاپ می‌شود. علت این است که پایتون عدد دریافت شده را به یک عبارت متنی تبدیل کرده و می‌دانیم ضرب یک عدد در یک عبارت متنی باعث می‌شود تا آن عبارت به تعداد عددی که در آن ضرب شده است، پشت سرهم تکرار شود. به همین دلیل ضرب عدد سه در عبارت 10 بجای عدد 30 عبارت 101010 را تولید کرده است.



یک مثال جالب دیگر :

```

a = input("لطفا عدد اول را وارد کنید: ")
b = input("لطفا عدد دوم را وارد کنید: ")
print(a + b)

```

قرار است a و b را با یکدیگر جمع کند

به جای عدد 50 عدد 2030 نوشته می‌شود

برنامه با استفاده از دستور `input()` دو عدد از کاربر دریافت می‌کند و در متغیرهای `a` و `b` قرار می‌دهد. اما هم زمان با این ذخیره سازی، اعداد را به عبارات **متنی** تبدیل می‌نماید. در خط بعد دستور `print ( )`، متغیرهای `a` و `b` با یکدیگر جمع می‌کند. اگر در این متغیرها اعداد 20 و 30 قرار داشت، باید عدد 50 چاپ می‌شد. اما در این متغیرها دو عبارت **متنی** قرار دارد که نمی‌توان آنها را با یکدیگر جمع ریاضی نمود.

```

>>> "ali" + "reza"
'alireza'
>>>

```

نکته: اگر در دو طرف عملگر جمع، دو داده از نوع **متنی** وجود داشته باشد، عملگر جمع، آنها را به یکدیگر متصل می‌نماید یعنی عبارت **متنی** جدیدی ایجاد می‌کند که از به هم پیوستن آن دو عبارت به وجود آمده است.

با توجه به این نکته می‌توان فهمید چرا در برنامه قبل، بعد از دریافت دو عدد 20 و 30 با استفاده از دستور `input()` و ذخیره آنها در متغیرهای `a` و `b`، جمع آنها با یکدیگر عدد 50 را تولید نکرده و بجای آن عبارت 2030 به چاپ رسیده است.

اما راه حل چیست؟ ما می‌خواهیم پایتون با داده‌های وارد شده، مثل یک عدد رفتار کند.

تابع `int()`

از این تابع برای تبدیل یک داده‌ی **متنی** به داده‌ی **عددی** از نوع صحیح استفاده می‌شود. به

```

>>> 3 + int("5")
8
>>>

```

مثال زیر توجه نمایید. مشاهده می‌کنید که عبارت **متنی** "5" داخل تابع `int()` قرار داده شده است. این تابع با تبدیل آن به عدد 5 و جمع کردن آن با عدد 3، عدد 8 را تولید نموده است.



اگر در مثال قبل از تابع `int()` استفاده نمی‌کردیم، چنین مشکلی پیش می‌آمد:

در این خط یک عدد صحیح را می‌خواهیم با یک عبارت متنی جمع کنیم که امکان آن وجود ندارد به همین خاطر پیغام خطا ظاهر شده است.

خطوطی از پیغام خطا

محو شده است که برنامه خوانا تر باشد.

```
>>> 3 + "5"
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    3 + "5"
TypeError: unsupported operand type(s) for +: 'int' and 'str'

>>> 3 + int("5")
8
```

در این خط ابتدا عبارت متنی "5" را با استفاده از تابع `int()` به داده‌ی عددی تبدیل می‌کنیم.

تبدیل ورودی دستور `input()` به داده‌ی عددی از نوع صحیح

اگر بخواهیم عبارت دریافت شده از دستور `input()` را به یک داده‌ی عددی از نوع صحیح تبدیل نماییم، کافیست دستور `input()` را در داخل تابع `int()` قرار دهیم.

به مثال زیر توجه نمایید:

عبارتی که دستور `input()` دریافت می‌کند را به داده‌ی عددی از نوع صحیح تبدیل می‌نماید.

```
a = int(input("Enter a number : "))
print( 3 * a )
```

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help

Enter a number : 10
30
>>>
```

توجه داشته باشید که اگر از تابع `int()` استفاده می‌کنید، حتماً در زمان اجرای برنامه باید یک عدد صحیح به کامپیوتر وارد نمایید وگرنه با خطای پایتون مواجه خواهید شد. در غیر اینصورت یعنی وارد کردن یک عدد اعشاری باعث ظاهر شدن خطا می‌شود. به عنوان مثال

اگر در زمان اجرای برنامه‌ی فوق، عدد 10.5 که یک عدد اعشاری است را وارد نمایید، پیغام مقابل ظاهر می‌شود:

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help

Enter a number : 10.5
Traceback (most recent call last):
  File "C:\Club Python\Programs\Pro\08.py", line 2, in <module>
    a = int(input("Enter a number : "))
ValueError: invalid literal for int() with base 10: '10.5'
```



دست گرمی اول

به نظر شما اگر در زمان اجرای برنامه روبرو کاربر عدد 8 را وارد نماید، خروجی آن چه خواهد بود؟

یک اشتباه بسیار رایج

در دستوراتی که در آنها از چند پرانتز تو در تو استفاده می‌شود باید توجه داشت که تعداد

```

a = input("Enter a number : ")
print( int(a) * a )
  
```

پرانتزهای باز و بسته با یکدیگر برابر باشند. در غیر این صورت اجرای برنامه با مشکل مواجه می‌شود. مانند مثال مقابل که کاربر در سطر اول دو پرانتز باز کرده اما تنها یک پرانتز را بسته است.

با اجرای این برنامه یک پنجره کوچک مطابق شکل زیر ظاهر می‌شود و قسمتی از برنامه که دارای اشکال است، با رنگ مشخص

```

a = int(input("Enter a number : "))
print( 3 * a )
  
```

می‌شود. در حقیقت چون پرانتز

آخر بسته نشده است، سایر دستورات را ادامه دستور قبل می‌داند و در نتیجه پیغام خطا ظاهر می‌گردد.

دست گرمی دوم

برنامه‌ای بنویسید که یک حرف یا کاراکتر و نیز یک عدد را از کاربر دریافت نماید و آن کاراکتر را به تعداد عدد دریافت شده در یک سطر چاپ نماید. مثلاً اگر کاربر حرف "*" و عدد 5 را وارد نماید، در خروجی پنج ستاره مطابق شکل مقابل چاپ شود.

```

>>>
Enter a chracter : *
Enter a number :5
*****
>>>
  
```

تابع `float()`

بالتر اشاره شد که قرار دادن یک عدد اعشاری داخل تابع `int()` قابل قبول نیست. پایتون برای این کار تابعی به نام `float()` دارد که وظیفه اش این است تا یک داده متنی را به داده‌ی عددی از نوع اعشاری تبدیل کند. به مثال مقابل توجه نمایید. مشاهده می‌کنید که عبارت متنی "4.5" داخل تابع `float()` قرار داده شده است. این تابع با تبدیل آن به عدد اعشاری 4.5 و ضرب آن در عدد ۲، عدد 9.0 را تولید نموده است.

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>> 2 * float("4.5")
9.0
>>>
```

تبدیل ورودی دستور `input()` به داده‌ی عددی از نوع اعشاری

اگر بخواهیم عبارت دریافت شده از دستور `input()` را به یک داده‌ی عددی از نوع اعشاری تبدیل نماییم، کفایت خروجی دستور `input()` را داخل تابع `float()` قرار دهیم. به مثال زیر توجه کنید:

داده‌ای را از کاربر دریافت کرده و آن را به صورت داده‌ی متنی در متغیر a ذخیره می‌کند

```
a = input("Enter a number : ")
b = float(a)
print(2 + b)
```

داده‌ی متنی موجود در متغیر a را به داده‌ی عددی از نوع اعشاری تبدیل می‌کند

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>>
Enter a number : 3.5
5.5
>>>
```

برنامه فوق را به صورت خلاصه‌تر نیز می‌توان نوشت. یعنی به صورت مستقیم دستور `input()` را داخل دستور `float()` قرار داد.

```
15.py
File Edit Format Run Options Window Help
a = float(input("Enter a number : "))
print(2 + a)
```

دستور `input()` مستقیم در داخل دستور `float()` قرار داده شده است.



برتری دستور `float()` نسبت به `int()`

همانطور که قبلاً گفتیم با استفاده از دستور `int()` تنها می‌توان داده‌های عددی از نوع صحیح ایجاد نمود و اگر یک دادهٔ متنی مانند "4.5" را در داخل آن قرار دهیم با پیغام خطا مواجه خواهیم شد.

از طرفی گفتیم که با استفاده از دستور `float()` می‌توان داده‌های عددی از نوع اعشاری ایجاد نمود. اما تفاوت آن با تابع `int()` در این است که اگر یک دادهٔ متنی مانند "4" را در داخل آن قرار دهیم، پیغام خطا ظاهر نمی‌شود. به مثال زیر توجه نمایید.

```
Python 3.6.6 Shell
File Edit Shell Debug Options Window Help
>>> float("4")
4.0
>>>
```

بله همانطور که مشاهده می‌کنید، باز هم یک عدد اعشاری تولید شده است. از نظر پایتون عدد 4 یک عدد صحیح و عدد 4.0 یک عدد اعشاری به حساب می‌آید. اما از آنجا که مقدار عددی 4 با 4.0 برای ما تفاوتی ندارد، هیچ مشکلی در حل مسائل ایجاد نمی‌شود.

به همین دلیل معمولاً از دستور `float()` برای تبدیل ورودی دستور `input()` استفاده می‌شود تا هم بتوان اعداد صحیح و هم اعداد اعشاری را در زمان اجرای برنامه وارد نمود.

آشنایی با Comment

به دلایلی که آنها را بررسی خواهیم کرد، برنامه نویسان عباراتی را در داخل برنامه خود می‌نویسند که نمی‌خواهند آن عبارات اجرا شوند. به این عبارات، توضیح یا Comment گفته می‌شود.

دلایل استفاده از Comment

۱- در بعضی از موارد لازم است توضیحات یا اطلاعاتی در رابطه با برنامه، داخل خود برنامه نوشته شود تا در صورت مراجعه به آنها در آینده، ابهامی در ارتباط با دستورات و نحوهٔ اجرای آنها برای ما یا کسانی که برنامه ما را می‌خوانند، وجود نداشته باشد. این اطلاعات ممکن است اطلاعاتی در رابطه با شخص برنامه نویس یا عملکرد یک دستور و یا هر چیزی باشد که برنامه نویس تشخیص می‌دهد وجود این اطلاعات در آینده ممکن است مفید باشد.



۲- در بعضی از موارد دستوری در برنامه نوشته شده است که به صورت موقت نمی‌خواهیم اجرا شود. این دستور را به حالت Comment در می‌آوریم تا پایتون آن را اجرا نکند. این عمل معمولاً در زمان **Debug** کردن یعنی رفع اشکالات موجود در برنامه مورد استفاده قرار می‌گیرد.

نحوه معرفی Comment

در زبان برنامه نویسی پایتون، خطی که با علامت # شروع شود، Comment شناخته شده و کامپیوتر آن را اجرا نمی‌کند. البته لازم نیست این علامت حتماً در ابتدای خط قرار گیرد. یعنی گاهی لازم نیست کل خط Comment باشد. اگر علامت # در وسط یک خط قرار گیرد، از آنجای خط به بعد، Comment محسوب می‌شود. مانند مثال زیر که در سطر دوم، تنها قسمتی از این سطر Comment شده است درحالی که در قسمت سمت چپ آن دستوری وجود دارد که Comment نیست و اجرا می‌شود.

```

13.py
File Edit Format Run Options Window Help

# لطفا در زمان اجرای برنامه یک عدد صحیح وارد نمایید
a = int(input("Enter a number : "))
print( 3 * a )

Ln: 4 Col: 38

```

دست گرمی اول رو انجام دادم یه عالمه 8 چاپ شد. درسته؟

88888888



رو دست گرمی دوم هم فکر کردم اما درست نتونستم حلش کنم. اگه تو می‌تونی حلش کن و در تالار پرسش و پاسخ به من هم بگو!