برنامه نویس هستید؟ پایتون را نیز از اینجا یاد بگیرید.

اگر قصد دارید که زبان برنامه نویسی تازه ای یاد بگیرید، پایتون گزینه خوبی است. نه به این خاطر که یادگیری اش آسان است بلکه به خاطر کاربردهای عملی که دارد و در رسیدن به بسیاری از اهداف ما را یاری می کند.

این مقاله مخصوص افرادی است که تجربه برنامه نویسی داشته اند و می خواهند به سرعت پایتون یاد بگیرند. اگر تا به حال تجربه برنامه نویسی نداشته اید، پیشنهاد می کنیم ابتدا در سایت های آموزش آنلاین، مقدمات پایتون را یاد بگیرید.

تمام مثال هایی که در ادامه آورده شده است از Python 3.x است و تضمینی نمی شود که در Python 2.x هم کار کند اما محتوا و اصول یکی است.

Strings

دستکاری String چیزی است که همه برنامه نویسان پایتون باید بلد باشند. String در همه زمینه ها استفاده می شود، از برنامه نویسی وب گرفته تا برنامه نویسی بازی و تجزیه تحلیل داده ها. یک راه درست و یک راه غلط برای کار کردن با String ها در پایتون وجود دارد.

String Formatting

فرض کنید ۲ رشته داریم:

```
>>>name = "Joel"
>>>job = "Programmer"
```

و می خواهید که این ۲ رشته را به هم بچسبانید تا یکی شود. اکثر افراد تمایل دارند این کار را اینگونه انجام دهند:

```
>>>title = name + " the " + job
>>>title
>"Joel the Programmer"
```

اما این روش در پایتون صحیح نیست. راه سریع تری وجود دارد تا با دستکاری رشته ها، کدهای خوانا تری داشته باشیم، با استفاده از روش format():

```
>>>title = "{} the {}".format(name, job)
>>>title
>"Joel the Programmer"
```

{ } جایگزین پارامترهای Format می شود. اولین { } جایگزین name و دومین { } جایگزین job شده است. هر چقدر که { } یا پارامتر که لازم داشته باشید می توانید استفاده کنید، فقط تعدادشان باید برابر باشد.

پارامترها می توانند که رشته نباشند. هرچیزی می تواند در قالب رشته بیان شود، مثل اعداد صحیح:

```
>>>age = 28
>>>title = "{} the {} of {} years".format(name, job, age)
>>>title
>"Joel the Programmer of 28 years"
```

String Joining

یکی دیگر از ترفندهای جذاب پایتون join() است که تعدادی رشته را می گیرد و آنها را با هم ترکیب کرده و یکی می کند. مانند زیر:

```
>>>availability = ["Monday", "Wednesday", "Friday", "Saturday"]
>>>result = " - ".join(availability)
>>>result
>'Monday - Wednesday - Friday - Saturday'
```

خروجی عبارت بالا، قرار گرفتن dash بین آیتم های String می باشد. این بدین معنی است که dash اضافه در انتهای String ظاهر نخواهد شد. استفاده از روش join خیلی سریعتر از روش دستی است.

Conditionals

برنامه نویسی بدون شرط خیلی بیهوده است. خوشبختانه، شرط ها در زبان پایتون بسیار شفاف و ساده هستند و آدم را گیج نمی کنند. در ادامه می بینیم که چطور پایتون به زیبایی این کار را انجام می دهد.

Boolean Values

مانند تمام زبان های دیگر برنامه نویسی، عملگرهای مقایسه ای یک نتیجه منطقی را ارزیابی می کنند. مانند صحیح و غلط. در ادامه چند عملگر مقایسه ای در پایتون را می بینیم:

```
>>>x = 10
>>>print(x == 10) # True
>>>print(x != 10) # False
>>>print(x <> 10) # False, same as != operator
>>>print(x > 5) # True
>>>print(x < 15) # True
>>>print(x >= 10) # True
>>>print(x <= 10) # True
```

The is and not Operators

عملگرهای ==، =! و <> که در بالا آمده اند، برای مقایسه ارزش ۲ متغیر بکار رفته اند. اگر می خواهید بفهمید که ۲ متغیر به یک مورد اشاره می کنند یا نه از عملگر is استفاده کنید:

```
>>>a = [1,2,3]
>>>b = [1,2,3]
>>>c = a
>>>print(a == b) # True
>>>print(a is b) # False
>>>print(a is c) # True
```

با عملگر not می توانید نتیجه منفی یک گزاره منطقی را بیان کنید، اینگونه که not را قبل آن بکار ببرید:

```
>>>a = [1,2,3]
>>>b = [1,2,3]
>>>if a is not b:
>>>    # Do something here
>>>x = False
>>>if not x:
>>>    # Do something here
```

The in Operator

اگر می خواهید چک کنید که آیا مقداری در object های تکرارشونده وجود دارد مثل لیست و Dict، سریع ترین راه استفاده از عملگر in است:

```
>>>availability = ["Monday", "Tuesday", "Friday"]
>>>request = "Saturday"
>>>if request in availability:
>>>    print("I'm available on that day!")
```

Complex Conditionals

می توانید چندین عبارت شرطی را به کمک عملگرهای and و or باهم ترکیب نمایید. عملگر and اگر در دو طرف آن مقادیر صحیح باشد، مقدار صحیح را بر می گرداند، و اگر یکی از طرفین غلط باشد، غلط را بر می گرداند. عملگر or اینگونه است که اگر یک طرف آن صحیح باشد، صحیح را بر می گرداند در غیر این صورت غلط را بر می گرداند.

```
>>>legs = 8
>>>habitat = "Land"
>>>if legs == 8 and habitat == "Land":
>>>    species = "Spider"
```

```
>>>weather = "Sunny"
>>>if weather == "Rain" or weather == "Snow":
>>>    umbrella = True
>>>else:
>>>    umbrella = False
```

می توانید خلاصه تر نیز ازآنها استفاده کنید:

```
>>>weather = "Sunny"
>>>umbrella = weather == "Rain" or weather == "Snow"
>>>umbrella
>False
```

Loops

پایه ای ترین حلقه در پایتون، حلقه While است، که تا زمانی که شرط برقرار باشد، کار می کند:

```
>>>i = 0
>>>while i < 10:
>>>    print(i)
>>>    i = i + 1
```

این گونه هم می توان نوشت:

```
>>>i = 0
>>>while True:
>>>    print(i)
>>>    if i >= 10:
>>>    break
```

گزاره Break برای خروج فوری از حلقه استفاده می شود. اگر می خواهید که از ادامه این حلقه دست بردارید و تکرار بعدی را آغاز کنید، می توانید از Continue استفاده نمایید.

The For Loop

مهم ترین حلقه ای که با پایتون سازگار است حلقه for است. حلقه for در پایتون با حلقه for که در زبان های دیگر مثل C، C# و Java می شناختید، متفاوت است. این حلقه بیشتر شبیه به foreach در آن زبان ها است.

حلقه for، به کمک عملگر in، object های تکرارشونده را تکرار می کند.

```
>>>weekdays = ["Monday", "Tuesday", "Wednesday",
"Thursday", "Friday"]
>>>for day in weekdays:
>>>    print(day)
```

حلقه for کار خود را از ابتدای لیست Weekday آغاز می کند، اولین مورد از لیست Weekday را به پارامتر day اختصاص می دهد و با این کار وارد چرخه اول حلقه for می شود. زمانی که اولین حلقه تمام شد، پارامتر بعدی از لیست Weekday را به day اختصاص می دهد، و به همین ترتیب حلقه ادامه دار می شود تا به آخرین مورد از لیست Weekday برسد.

اگر می خواهید در پایتون برای تعداد زیادی تکرار حلقه بنویسید، از range() کمک بگیرید:

```
>>># Prints 0,1,2,3,4,5,6,7,8,9
>>>for i in range(10):
>>>    print(i)
```

زمانی که range() تنها یک پارامتر دارد، از صفر شروع می کند و یکی یکی جلو می رود، اما اگر دو پارامتر به آن بدهید، با مقدار اولی شروع می کند و یکی یکی جلو می رود ولی تا قبل از عدد آخر می ایستد.

```
>>># Prints 5,6,7,8,9
>>>for i in range(5, 10):
>>>    print(i)
```

اگر می خواهید که با فاصله بشمرد نه یکی یکی، می توانید پارامتر سوم را نیز به آن اضافه کنید که آن میزان فاصله است. مثال زیر همان مثال قبل است فقط یکی یکی نمی شمرد:

```
>>># Prints 5,7,9
>>>for i in range(5, 10, 2):
>>>    print(i)
```

Enumerations

اگر از زبان دیگری به سراغ پایتون می آیید، باید حواستان باشد که حلقه زدن روی Object های تکرار شونده، index آن Object را در لیست به شما نمی دهد. Index ها در پایتون معنایی ندارند و نباید از آنها استفاده کرد ولی اگر خیلی تمایل به استفاده از آن مفهوم دارید می توانید از روش enumerate() استفاده کنید:

```
>>>weekdays = ["Monday", "Tuesday", "Wednesday",
"Thursday", "Friday"]
>>>for i, day in enumerate(weekdays):
>>>    print("{} is weekday {}".format(day, i))
```

نتیجه اش به شرح زیر خواهد بود:

```
>Monday is weekday 0
>Tuesday is weekday 1
>Wednesday is weekday 2
>Thursday is weekday 3
>Friday is weekday 4
```

در مقایسه با سایر زبان ها، پایتون مانند آنها رفتار نمی کند:

```
>>>i = 0
>>>for day in weekdays:
>>>    print("{} is weekday {}".format(day, i))
>>>    i = i + 1
```

Dictionaries

Dictionary یا Dict یکی از مهم ترین Data type های موجود در زبان پایتون است. همیشه از آنها استفاده می کنید. بکارگیری آنها آسان است، سریع عمل می کنند و کدنویسی شما را تمیز و مرتب می کنند.

با ماهر شدن در Dict ها نیمی از چالش یادگیری پایتون را پشت سر گذاشته اید.

نکته جالب در مورد Dict ها این است که ازآنها استفاده می کنید ولی با این نام نمی شناختید، شاید بیشتر به نام های hash tables وhash maps از آنها استفاده می کردید. در واقع آرایه های پیوندی از مقادیر ۲تایی Key و Value هستند. در لیست به کمک index می توانید به محتوای هر خانه آرایه دست پیدا کنید، در Dict این وظیفه بر گردن key است.

چطور یک Dict خالی را نمایش دهیم:

```
>>>d = {}
```

چطور یک Dict Key را به یک Value اختصاص دهیم:

```
>>>d = {}
>>>d["one_key"] = 10
>>>d["two_key"] = 25
>>>d["another_key"] = "Whatever you want"
```

نکته جالب توجه در مورد Dict این است که می توانید انواع متغیر ها را باهم ترکیب نمایید. در واقع اصلا مهم نیست که چه چیزی در درون آن قرار می دهید. برای استفاده راحت تر از Dict می توانید از ساختار زیر استفاده نمایید:

```
>>>d = {
>>>    "one_key": 10,
>>>    "two_key": 25,
>>>    "another_key": "Whatever you want"
>>>}
```

برای دسترسی به مقادیر موجود در Dict توسط Key از کد زیر استفاده نمایید:

```
>>>d["one_key"]
>10
>>>d["another_key"]
>"Whatever you want"
>>>d["one_key"] + d["two_key"]
>35
```

برای تکرار در Dict از حلقه for به شکل زیر استفاده نمایید:

```
>>>for key in d:
>>>    print(key)
```

برای تکرار کردن مقادیر و key می توانید از روش items() استفاده نمایید:

```
>>>for key, value in d.items():
>>>    print(key, value)
```

اگر می خواهید موردی را از درون Dict خارج نمایید از عملگر del استفاده نمایید:

```
>>>del d["one_key"]
```

Dict برای موارد متنوعی استفاده می شود. در ادامه یک مثال کوتاه از استفاده Dict را با هم می بینیم: نگاشت یا Mapping هر ایالت آمریکا به مراکز آنها. با استفاده از Dict اینگونه خواهد شد:

```
>>>capitals = {
>>>    "Alabama": "Montgomery",
>>>    "Alaska": "Juneau",
>>>    "Arizona": "Phoenix",
>>>    ...
>>>}
```

و هروقت که به مرکز هر ایالت نیاز داشتید، می توانید اینگونه به آن دسترسی پیدا کنید:

```
>>>state = "Pennsylvania"
>>>capitals[state]
>"Harrisburg"
```

یادگیری مداوم پایتون ارزشمند است

این موارد فقط مباحث پایه ای بودند که این زبان را از سایر زبان ها متفاوت می کرد. اگر مطالب موجود در این مقاله را فهمیدید، پس دقیقا در راه خبره شدن در زبان پایتون قرار دارید. این راه را ادامه دهید و هیچ زمانی را از دست ندهید.

اگر در دنبال کردن آن به مشکل خوردید، نگران نشوید. این اصلا به این معنی نسیت که برای برنامه نویسی ساخته نشده اید. تنها بدین معناست که یادگیری پایتون به آن سادگی هم که فکر می کردید نیست.

همه ما ممکن است در این راه به چالش کشیده شویم، پس نباید استرس بگیرید. اگر استرس گرفتید پس مقاله ما با عنوان: برای یادگیری برنامه نویسی استرس دارید؟ نگران نباشید، کلیک کنید، را بخوانید.