CS50x Iran

Conditionals

Week note

پاییز و زمستان ۱۴۰۴

1 فهرست

عبارت های شرطی به شما به عنوان برنامه نویس این امکان را می‌دهند که به برنامه ای که می‌نویسید قدرت تصمیم گیری بدهید. انگار که برنامه شما بین دو جاده ای که در سمت راست و چپ قرار دارد، بنابر عوامل مختلف حق انتخاب دارد.

در پایتون به صورت پیش فرض عملگر هایی وجود دارد که امکان پرسیدن سوال های ریاضی را به شما می‌دهند:

- دو عملگر > و < به معنای کوچکتر یا بزرگتر.
- عملگر =< به معنی بزرگتر و یا مساوی.
- عملگر => به معنی کوچکتر و یا مساوی.
- عملگر == به معنی برابری است، به دو علامت مساوی توجه کنید! در زبان های برنامه نویسی یک مساوی به معنای مقداردهی می‌باشد و دو مساوی برای مقایسه دو مقدار استفاده می‌شود.
- عملگر =! به معنی عدم برابری است.

دستورات شرطی همیشه عبارت سمت چپ را با عبارت سمت راست مقایسه می‌کنند!

در پنجره ترمینال خود دستور `code compare.py` را وارد کنید. این دستور یک فایل جدید به اسم compare ایجاد می‌کند. سپس کدهای زیر را وارد این فایل کنید:

```
compare.py

x = int(input("What's x? "))
y = int(input("What's y? "))

if x < y:
    print("x is less than y")
```

دقت کنید که چطور برنامه شما در ابتدا از کاربر دو مقدار x و y را دریافت می‌کند. سپس آن ها را تبدیل به عدد کرده و در مرحله بعد مقدار عدد بدست آمده را در متغیرهای x و y ذخیره می‌کند. سپس دستور `if` دو متغیر x و y را با هم مقایسه می‌کند و اگر شرط x < y درست باشد و یا به عبارتی دیگر، مقدار x از مقدار y کوچکتر باشد، دستور `print` اجرا شده و جمله‌ی x is less than y را چاپ می‌کند.

عبارت شرطی از مقدارهای boolean (که تنها شامل دو مقدار درست True و یا غلط False می‌باشد) برای تصمیم گیری درباره اجرا استفاده می‌کند. اگر شرط x < y برقرار باشد، حاصل این عبارت برابر با True بوده و به همین دلیل کدهای داخل شرط اجرا می‌شود.

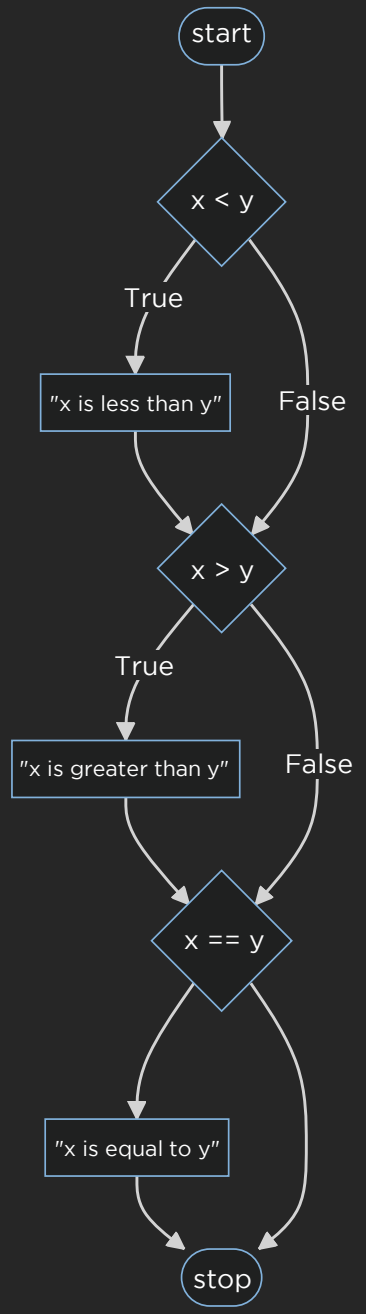
حالا کد خودتان را به شکل زیر ویرایش کنید:

compare.py

```
x = int(input("What's x? "))
y = int(input("What's y? "))

if x < y:
    print("x is less than y")
if x > y:
    print("x is greater than y")
if x == y:
    print("x is equal to y")
```

اگر دقت کنید، می‌بینید که از ۳ دستور if پشت سر هم استفاده کردیم. به ترتیب اول از همه if اول اجرا شده و شرط آن مورد بررسی قرار می‌گیرد. سپس نوبت if دوم و در آخر if سوم مورد بررسی قرار گرفته و اجرا می‌شود. این جریان تصمیم گیری را Control Flow می‌گویند.



کد صفحه‌ی قبل را می‌توان به شکل رو به رو نشان داد:

اما ما می‌توانیم برنامه‌ای که داریم را به شکل بهینه تری بنویسیم. به این صورت که ۳ سوال متوالی یکسان نپرسیم چون ممکن است که هر سه سوال همیشه درست نباشند که در صفحه ی بعد به این مورد می پردازیم.

6 کنترل روند برنامه با else، if، elif

برنامه‌ی خود را به صورت زیر اجرا کنید:

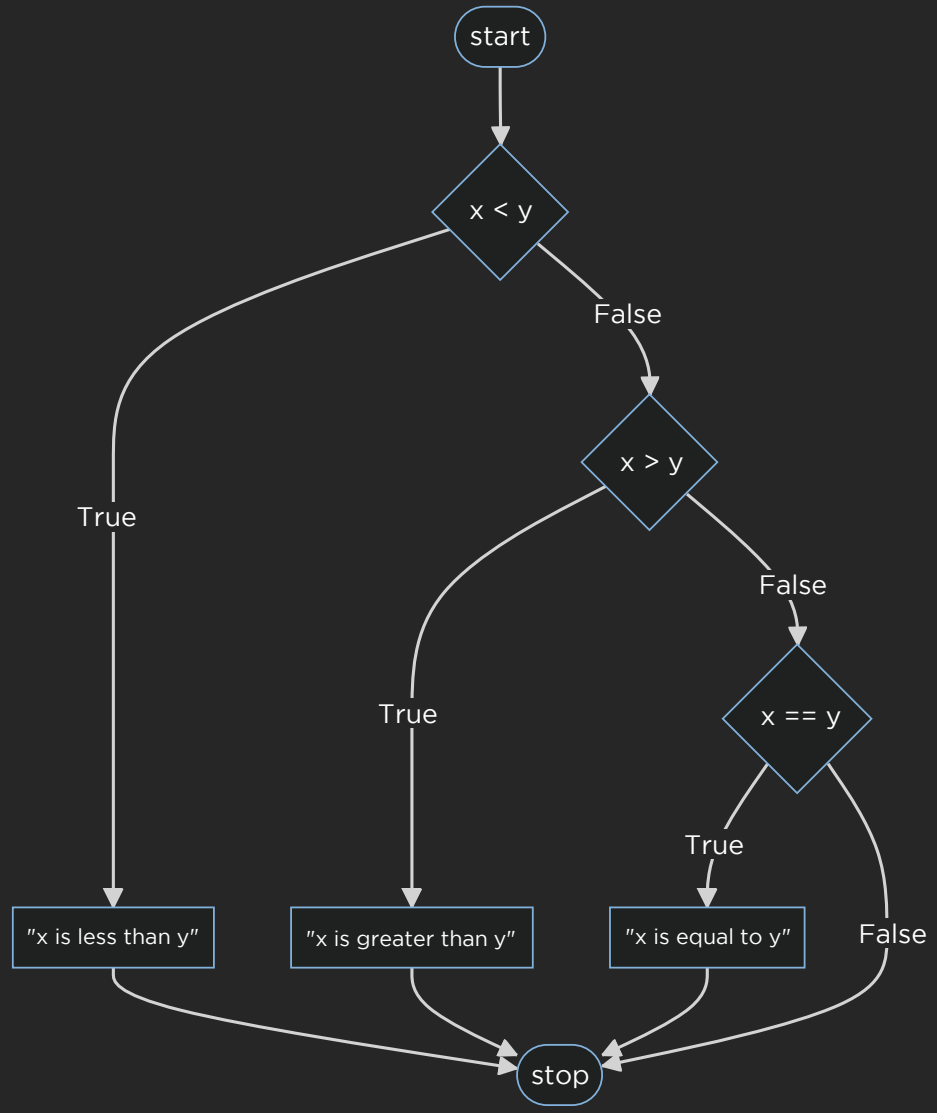
compare.py

```
x = int(input("What's x? "))
y = int(input("What's y? "))

if x < y:
    print("x is less than y")
elif x > y:
    print("x is greater than y")
elif x == y:
    print("x is equal to y")
```

به این نکته که استفاده از elif (به معنای "در غیر این صورت اگر") بجای if، باعث شده تا در طول اجرای برنامه تصمیم های کمتری گرفته شود توجه کنید.

اول از همه عبارت if بررسی می‌شود. اگر این عبارت درست باشد، دیگر برنامه به سراغ دو elif در ادامه نمی‌رود و این عبارت ها اصلا اجرا نمی‌شوند. اگر عبارت if نادرست باشد، برنامه سراغ elif اول می‌رود و اگر نتیجه‌ی elif درست باشد، برنامه بدون اجرا کردن elif دوم به پایان می‌رسد.



کد صفحه‌ی قبل را می‌توان به شکل رو به رو نشان داد:

ممکن است کامپیوتر شما تغییری در زمان و سرعت اجرا شدن دو برنامه احساس نکند، اما این نکته را در نظر بگیرید که برای یک سرور آنلاین که هر روز میلیاردها یا تیلیاردها از این نوع برنامه ها را اجرا می‌کند، این تصمیم کوچک در برنامه نویسی می‌تواند نکته قابل تاملی باشد.

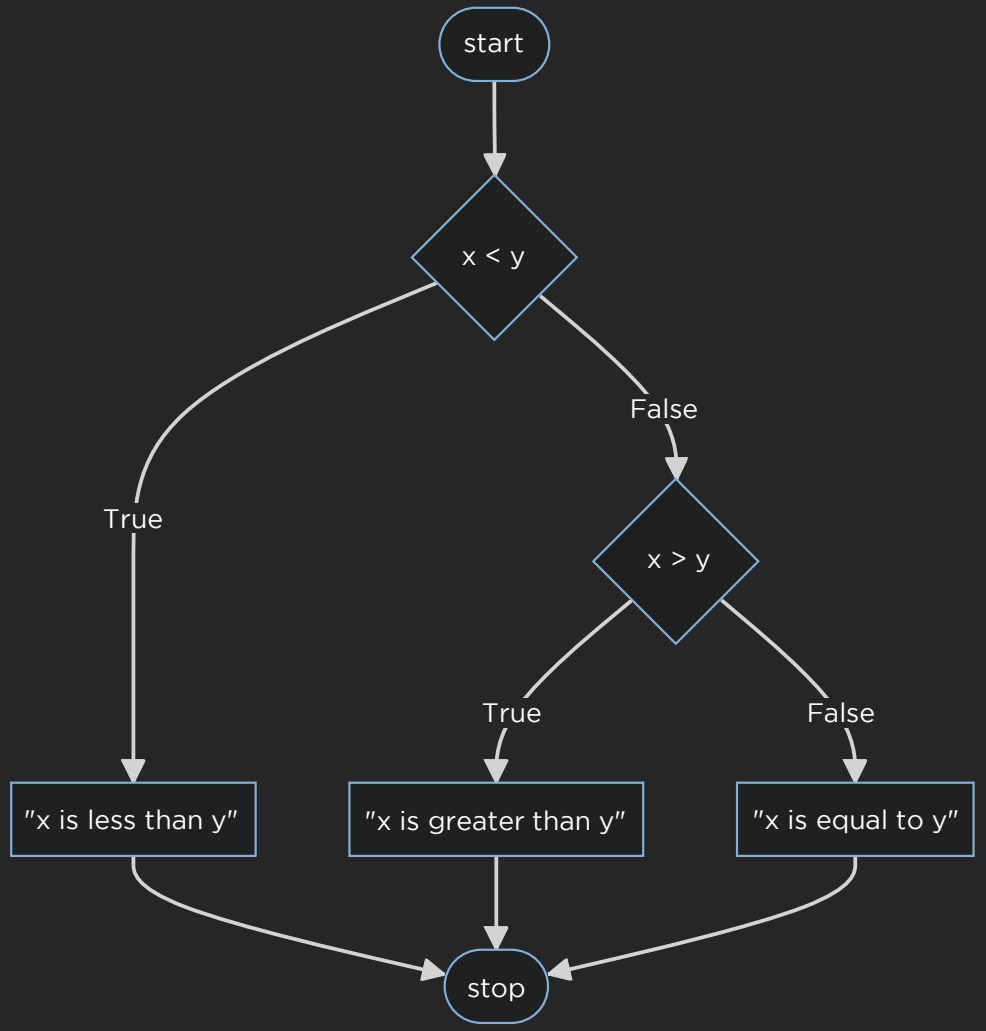
هنوز یک نکته کوچک دیگر هست که می‌تواند به ما در بهینه سازی برنامه کمک کند.

اگر به شرط elif سوم دقت کنید، متوجه می‌شوید که این شرط برای اجرای این برنامه خیلی ضروری نیست. با توجه به دو عبارت بالا، اگر به صورت منطقی مقدار x از y کوچکتر نباشد و همچنان x از y هم بزرگ تر نباشد (دو شرط بالا مقدار False داشته باشند)، نتیجه می‌گیریم x با y مقدار یکسانی دارد. با جایگزینی یک else به معنای (اگر همه حالات False بود)، می‌توانیم کد خود را به صورت زیر اجرا کنیم:

compare.py

```
x = int(input("What's x? "))
y = int(input("What's y? "))

if x < y:
    print("x is less than y")
elif x > y:
    print("x is greater than y")
else:
    print("x is equal to y")
```



توجه کنید که چگونه از طریق مروری که تا اینجا روی برنامه خود داشتیم، پیچیدگی آن نسبت به حالت اولیه، کاهش پیدا کرده است.

کد نهایی خود را می‌توانیم به شکل رو به رو نمایش دهیم:

استفاده از or (به معنای "یا") به برنامه شما امکان انتخاب بین یک یا چند گزینه را می‌دهد. به عنوان مثال، ما می‌توانیم برنامه خود را به صورت زیر ویرایش کنیم:

compare.py

```
x = int(input("What's x? "))
y = int(input("What's y? "))

if x < y or x > y:
    print("x is not equal to y")
else:
    print("x is equal to y")
```

ترجمه: x با y برابر نیست

توجه داشته باشید که نتیجه برنامه تغییری نکرده است، اما پیچیدگی آن کاهش و کارایی آن افزایش یافته است. در این مرحله کد ما بسیار عالی است. با این حال آیا می‌توان طراحی را بهبود بیشتری بخشید؟ به نکات صفحه‌ی بعد توجه داشته باشید.

بیایید کد خود را به صورت روبرو ویرایش کنیم:

compare.py

```
x = int(input("What's x? "))
y = int(input("What's y? "))

if x != y:
    print("x is not equal to y")
else:
    print("x is equal to y")
```

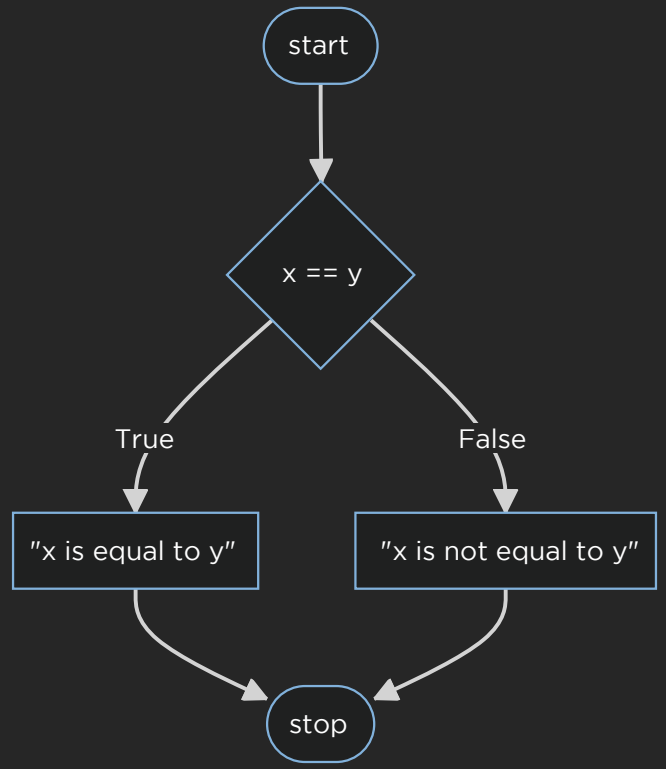
توجه کنید که چگونه or را به طور کامل حذف کردیم و به سادگی پرسیدیم " آیا x برابر با y نیست؟ ". بنابراین ما فقط و فقط یک سوال می‌پرسیم. بسیار کارآمد! و یا می‌توانیم کد خود را به صورتی که در صفحه بعد اشاره شده تغییر دهیم.

دقت کنید که عملگر == بررسی می‌کند که آیا آنچه در سمت چپ و راست آمده است با یکدیگر برابر هستند یا خیر. استفاده از دو علامت مساوی بسیار مهم است. اگر فقط از یک علامت مساوی استفاده کنید، احتمالا توسط مفسر پایتون، خطایی ایجاد می‌شود. کد نهایی را می توان به صورت زیر نشان داد:

compare.py

```
x = int(input("What's x? "))
y = int(input("What's y? "))

if x == y:
    print("x is equal to y")
else:
    print("x is not equal to y")
```



درست مانند or از and (به معنای "و") هم می‌توان در عبارات شرطی استفاده کرد.

در پنجره ترمینال دستور code grade.py را اجرا کنید و برنامه جدید خود را به صورت زیر شروع کنید:

grade.py

```
score = int(input("Score: "))

if score >= 90 and score <= 100:
    print("Grade: A")
elif score >= 80 and score < 90:
    print("Grade: B")
elif score >= 70 and score < 80:
    print("Grade: C")
elif score >= 60 and score < 70:
    print("Grade: D")
else:
    print("Grade: F")
```

با اجرای python grade.py می‌توانید یک امتیاز وارد کنید و یک نمره را به عنوان خروجی برنامه دریافت کنید. کد ما به درستی اجرا می‌شود، اما این کد در برابر مشکلات احتمالی آسیب پذیر است.

به طور معمول، ما هرگز نمی‌خواهیم به کاربر خود برای وارد کردن اطلاعات صحیح اعتماد کنیم. به این منظور کد خود را به شکل زیر ویرایش می‌کنیم:

grade.py

```
score = int(input("Score: "))

if 90 <= score <= 100:
    print("Grade: A")
elif 80 <= score < 90:
    print("Grade: B")
elif 70 <= score < 80:
    print("Grade: C")
elif 60 <= score < 70:
    print("Grade: D")
else:
    print("Grade: F")
```

می‌بینید که چگونه پایتون این امکان را به شما می‌دهد تا عملگر ها و شرط ها را طوری در کنار هم قرار دهید که در سایر زبان های برنامه نویسی روشی غیرمعمول و یا حتی غیرممکن باشد!

بیایید برای بار دیگر برنامه خود را بهبود ببخشیم:

grade.py

```
score = int(input("Score: "))

if score >= 90:
    print("Grade: A")
elif score >= 80:
    print("Grade: B")
elif score >= 70:
    print("Grade: C")
elif score >= 60:
    print("Grade: D")
else:
    print("Grade: F")
```

حالا می‌توانیم ببینیم که برنامه ما چطور با پرسیدن سوال های کمتر، توسعه می‌یابد.

این روش باعث می‌شود کد های ما بیشتر قابل خواندن باشد و در هنگام رفع مشکلات (به اصطلاح "باگ ها") به تلاش کمتری نیاز داشته باشد.

در ریاضیات، Parity به زوج و یا فرد بودن اعداد اشاره دارد.

عملگر % یا Modulo در برنامه نویسی به این منظور استفاده می‌شود تا بررسی کند آیا تقسیم دو عدد، دارای باقی مانده می‌باشد یا خیر.

برای مثال، حاصل عبارت 4 % 2 برابر با صفر خواهد بود، چون این دو عدد نسبت به هم بخشپذیر هستند و تقسیم آن ها دارای باقی مانده نمی‌باشد. اما حاصل عبارت 3 % 2 نتیجه ای متفاوت خواهد داشت. زیرا حاصل تقسیم این دو عدد، یک عدد صحیح نیست و دارای باقی مانده می‌باشد.

در ترمینال خود با وارد کردن دستور `code parity.py` یک برنامه جدید باز کنید و کدهای زیر را وارد آن کنید.

parity.py

```
x = int(input("What's x? "))

if x % 2 == 0:
    print("Even")
else:
    print("Odd")
```

ببینید که کاربران ما چطور این امکان را دارند تا با وارد کردن اعداد بزرگ تر از 1 متوجه بشوند که این عدد زوج و یا فرد است.

همانطور که در هفته صفر گفته شد، استفاده و ایجاد توابع در برنامه خود می‌تواند مفید واقع شود.

به این منظور، ما می‌توانیم تابعی در برنامه خود ایجاد کنیم که زوج و یا فرد بودن یک عدد را مشخص کند.

parity.py

```
def main():
    x = int(input("What's x? "))
    if is_even(x):
        print("Even")
    else:
        print("Odd")


def is_even(n):
    if n % 2 == 0:
        return True
    else:
        return False

main()
```

به این نکته توجه کنید که شرط اصلی ما که شامل تابع is_even می‌باشد.

چطور شرط بدون نوشتن هیچ عملگری کار می کند؟

دلیل این اتفاق این است که تابع is_even یک متغیر از نوع boolean را به عنوان خروجی بر می‌گرداند و در نهایت چیزی که برای دستور شرطی ما اهمیت دارد، حاصل عبارت مورد بررسی است که در اینجا در صورت زوج بودن عدد برابر با True و در صورت زوج نبودن برابر با False می‌باشد.

در دنیای برنامه نویسی، روش هایی وجود دارند که اصطلاحا به آن ها Pythonic گفته می‌شود. یعنی این روش ها تنها در زبان پایتون قابل استفاده و اجرا هستند. به عنوان مثال، کد ویرایش شده زیر را درنظر بگیرید.

با خواندن آن متوجه خواهید شد که دستور return تقریبا یک جمله به زبان انگلیسی است. این یک حالت خاص در کدنویسی می‌باشد که تنها در زبان پایتون دیده می‌شود.

parity.py

```
def main():
    x = int(input("What's x? "))
    if is_even(x):
        print("Even")
    else:
        print("Odd")


def is_even(n):
    return True if n % 2 == 0 else False

main()
```

حالا می‌توانیم با مرور این کد، آن را باز هم روان تر و بیشتر قابل خواندن کنیم:

این تابع در قسمت `return` حاصل عبارت `n % 2 == 0` را باز می‌گرداند که می‌تواند بین دو حالت True و یا False باشد. درنهایت این خروجی به تابع main ارسال می‌شود.

parity.py

```
def main():
    x = int(input("What's x? "))
    if is_even(x):
        print("Even")
    else:
        print("Odd")


def is_even(n):
    return n % 2 == 0

main()
```

همانند if، elif و else، دستور match هم برای بررسی شرط ها و کنترل برنامه مورد استفاده قرار می‌گیرد.

به عنوان مثال کد زیر را در نظر بگیرید:

house.py

```
name = input("What's your name? ")

if name == "Harry":
    print("Gryffindor")
elif name == "Hermione":
    print("Gryffindor")
elif name == "Ron":
    print("Gryffindor")
elif name == "Draco":
    print("Slytherin")
else:
    print("Who?")
```

در این کد سه شرط اول پاسخ یکسانی دارند. ما می‌توانیم برای بهبود و راحت تر کردن این کد، از عملگر or استفاده کنیم تا ظاهر کد خود را تمیزتر و خواناتر کنیم. به کد صفحه‌ی بعد توجه کنید:

house.py

```
name = input("What's your name? ")

if name == "Harry" or name == "Hermione" or name == "Ron":
    print("Gryffindor")
elif name == "Draco":
    print("Slytherin")
else:
    print("Who?")
```

house.py

```
name = input("What's your name? ")

match name:
    case "Harry":
        print("Gryffindor")
    case "Hermione":
        print("Gryffindor")
    case "Draco":
        print("Slytherin")
    case _:
        print("Who?")
```

می‌توانیم همین کد را با استفاده از دستور match پیاده کنیم:

دقت کنید که در آخرین case از علامت _ استفاده شده است. این علامت در case دقیقا همان رفتار else در if را دارد. دستور match مقداری که به آن داده شده است را به ترتیب با مقدار تمام دستورهای case مقایسه می‌کند و درصورت درست بودن شرط، کدهای داخل بلاک آن case اجرا خواهد شد و مقدار سایر case های دیگر مورد بررسی قرار نخواهد گرفت. می‌توان از این روش هم یک پله فرا تر رفت و بازهم کد را خواناتر کرد:

برای بررسی چندین شرط در case می‌توانیم از علامت | استفاده کنیم که دقیقا همان معنی or در if را می‌دهد.

house.py

```
name = input("What's your name? ")

match name:
    case "Harry" | "Hermione" | "Ron":
        print("Gryffindor")
    case "Draco":
        print("Slytherin")
    case _:
        print("Who?")
```

در این هفته مباحث زیر را مورد بررسی قرار دادیم:

- ✓ عبارت های شرطی
- ✓ دستور `if`
- ✓ کنترل روند برنامه با استفاده از `if` , `elif` , `else`
- ✓ عملگر `or`
- ✓ عملگر `and`
- ✓ عملگر %
- ✓ ایجاد توابع در کدهای خود
- ✓ نوشتن کدهای pythonic
- ✓ دستور `match`

حالا شما توانایی این را دارید که در پایتون از دستورات شرطی استفاده کنید و متناسب با نتیجه دریافتی، فرایند موردنظر خود را اجرا کنید.

CS50x Iran

Harvard's Computer Science 50x Iran