پایتون

جلسه هشتم

دکتر ایمان ذباح

با همکاری عرفان خوش صحبت

بهار ۱۴۰۳

فهرست مطالب

در مباحث شرط ها دیدیم که در صورتی که یک شرط درست باشد آن شرط اجرا خواهد شد و اگر نادرست باشد اجرا نخواهد شد. به این درستی و نادرستی True یا False می گوییم.

نکته: با استفاده از از دستور print و عملگر های شرطی (== و =! و => و ...) و عملگر های منطقی (and , or , not) می توانیم درست یا نادرست بودن یک عبارت را بسنجیم. مانند زیر:

```
print(7 >= 5)
```

خروجی:

```
True
```

مثال: درستی یا نادرستی عبارات زیر را با استفاده از دستور print چاپ کنید و خروجی را ملاحظه کنید.

```
8 != 2
6 == 2
12 > 6
2 < 2
```

```
print(8 != 2)
print(6 == 2)
print(12 > 6)
print(2 < 2)
```

خروجی:

```
True
False
True
False
```

نکته: با استفاده از تابع bool نیز میتوانیم درستی یا نادرستی عبارات را بسنجیم.

```
print(bool(8 >= 14))
print(bool(0))
print(bool(1))
```

خروجی:

```
False
False
True
```

نکته:

تقریبا هر مقداری که دارای یک نوع محتوا باشد مقدار true را برمیگرداند.

به جز رشته خالی(“”) ، هر رشته و یا متغیر رشته ای مقدار true را برمیگرداند.

هر عددی به جز عدد 0 مقدار true را برگشت می دهد.

همه listها، tupleها، setها و dictionaryها به جز مقدار خالی از آن ها مقدار true را برمیگرداند.

نکته: از این درستی (True) یا نادرستی (False) می توانیم در شرط ها استفاده هایی ببریم. و برنامه های بهینه تری بنویسیم.

مثال: به دو روش برنامه ای بنویسید که خالی بودن یا نبودن لیستی را چک کند و در خروجی چاپ کند.

روش اول:

```
mylist = []

if mylist:
    print("list item darad")
else:
    print("list itemi nadarad")
```

روش دوم:

```
mylist = []

if len(mylist) != 0:
    print("list item darad")
else:
    print("list itemi nadarad")
```

خروجی:

```
list itemi nadarad
```

مثال: برای مقادیر x و y به صورت ذهنی محاسبه کنید که حاصل عبارات زیر True خواهد شد یا False ؟

```
x = 5
y = 0
```

```
print(bool(y))
print(bool(x > 2 and y))
print(bool(x))
print(bool(not x))
print(bool(not y))
print(bool(x or y))
```

```
x = 5
y = 0
```

```
print(bool(y))
print(bool(x > 2 and y))
print(bool(x))
print(bool(not x))
print(bool(not y))
print(bool(x or y))
```

خروجی:

```
False
False
True
False
True
True
```

نکته: بولین ها نوعی از متغیر ها هستند که می توانند مقادیر True یا False را در خود ذخیره کنند. و همچنین به عنوان کلیدی برای شرط ها به کار بروند.

```
x = True
y = False
```

مثال: برنامه ای بنویسید که بدون استفاده از توابع داخلی پایتون (max, sort, ...) ماکسیمم 5 عدد دریافتی از کاربر را محاسبه کند. (برنامه باید برای اعداد منفی و اعداد مثبت بسیار بزرگ نیز کار کند.)

```python
numbers = []
for i in range(5):
    x = int(input(f"addad {i + 1}om ra vared kon: "))
    numbers.append(x)

ma = False
maximum = 0
for i in numbers:
    if ma:
        if i > maximum:
            maximum = i

    else:
        maximum = i
        ma = True

print("Maximum is: ", maximum)
```

نکته: در رشته ها همانند لیست ها می توانیم به هر خانه (index) مد نظر از رشته دسترسی داشته باشیم.

مثال : دومین (e) و آخرین کرکتر از رشته ی txt را چاپ کنید.

```
txt = "Hello World"
```

```
txt = "Hello World"
print("dovomin: ", txt[1])
print("akharin: ", txt[-1])
```

خروجی:

```
dovomin:  e
akharin:  d
```

نکته: برای گرفتن حروف بین دو خانه ی n و m از رشته ی txt می توانیم به شکل زیر عمل کنیم.

```
txt = "Hello World"
```

```
txt[m:n]
```

مثال : برنامه ای بنویسید که از خانه ی صفرم تا سوم از رشته ی txt را چاپ کند.

```
txt = "Hello World"
print(txt[0:3])
```

خروجی:

```
Hel
```

نکته: در نکته و مثال قبلی در صورتی که n را وارد نکنیم از حروف بین m الی آخرین حرف رشته را چاپ خواهد کرد.

```
txt = "Hello World"
```

```
txt[m:]
```

مثال : برنامه ای بنویسید که از خانه ی دوم تا آخر از رشته ی txt را چاپ کند.

```
txt = "Hello World"
print(txt[2:])
```

خروجی:

```
llo World
```

نکته: حال اگر m را وارد نکنیم حروف بین اولین حرف رشته تا n را چاپ خواهد کرد.

```
txt = "Hello World"
```

```
txt[:n]
```

مثال : برنامه ای بنویسید که هفت کرکتر اول از رشته ی txt را چاپ کند.

```
txt = "Hello World"
print(txt[:7])
```

خروجی:

```
Hello W
```

نکته: برای گرفتن حروف رشته با فاصله ی p از یکدیگر می توانیم به شکل زیر عمل کنیم.

```
txt = "Hello World"
```

```
[txt[::p]
```

مثال : برنامه ای بنویسید که حروف رشته تکست را به صورت یک درمیان چاپ کند.

```
txt = "Hello World"
print(txt[::2])
```

خروجی:

```
HloWrd
```

نکته: در صورتی که مقدار p را 1- بگذاریم رشته را وارون شده چاپ خواهد کرد.

```
txt = "Hello World"
```

```
txt = "Hello World"
print(txt[::-1])
```

خروجی:

```
dlroW olleH
```

نکته: متود های جدا کردن رشته که در نکات بالا دیدیم را متود اسلایسینگ (slicing) می گویند و می توان از آن همچنین برای لیست ها استفاده نمود.

نکته: با استفاده از تابع `join()` می توانیم اعضای یک لیست را به فرمت دلخواه به یکدیگر چسبانده و تبدیل به رشته (str) کنیم.

مثال: برنامه ای بنویسید که اعضای رشته ی mylist را به یکدیگر چسبانده و به متن تبدیل کند.

```
mylist = ['a', 'b', 'c', 'd', 'e']
```

```
mylist = ['a', 'b', 'c', 'd', 'e']
x = "".join(mylist)
print(x)
```

خروجی:

```
abcde
```

نکته: می توانیم با فرمت مدنظر خود آیتم های لیست را به یکدیگر بچسبانیم. مانند زیر:

```
x = "-".join(mylist)
print(x)
```

خروجی:

```
a-b-c-d-e
```

یا:

```
x = " ".join(mylist)
print(x)
```

خروجی:

```
a b c d e
```

یا:

```
x = "\n".join(mylist)
print(x)
```

خروجی:

```
a
b
c
d
e
```

نکته: در جلسات قبلی با کاربرد عملگر + برای رشته ها آشنا شدید. در این جلسه با کاربرد عملگر * برای رشته ها آشنا خواهید شد.

```
print("a" * 5)
```

خروجی کد بالا به شکل زیر چاپ خواهد شد:

```
aaaaa
```

مثال: برنامه ای بنویسید که تعداد ستاره های ردیف آخر را از کاربر گرفته و با توجه به آن شکل زیر را در خروجی چاپ کند.

```
*
**
***
****
*****
```

```
n = int(input("n ra vared kon: "))

txt = ""
for i in range(1, n + 1):
    txt = txt + (i * '*') + '\n'
print(txt)
```

```
n ra vared kon: 5
*
**
***
****
*****
```

خروجی:

نکته: با استفاده از تابع list() می توانیم سایر متغیر ها را به لیست تبدیل کنیم.

```
txt = 'hello'
print(list(txt))
```

```
['h', 'e', 'l', 'l', 'o']
```

همانطور که در نمونه ی بالا ملاحظه می کنید با استفاده از تابع list() برای رشته ها هر یک از کرکتر های رشته تبدیل به یک آیتم لیست شدند.

نکته: برای تبدیل اعداد صحیح(int) و اعشاری(float) به طور مستقیم نمی توانیم اقدام کنیم زیرا با ارور مواجه خواهیم شد. اما می توانیم از روش های جانبی استفاده کنیم.

برای مثال برای تبدیل ارقام یک عدد صحیح به آیتم های لیست میتوانیم ابتدا عدد را با استفاده از تابع str() به رشته تبدیل کنیم سپس با استفاده از حلقه و تابع int() به اعداد صحیح بازگردانی کنیم.

```
y = 348
y = str(y)
y = list(y)
for i in range(len(y)):
    y[i] = int(y[i])
```

خروجی:

```
[3, 4, 8]
```

نکته: با استفاده از تابع list() برای دیکشنری ها کلید (keys) های دیکشنری به آیتم های لیست تبدیل خواهند شد.

مباحث اسلایسینگ که برای رشته ها گفته شد برای لیست ها هم قابل استفاده است مانند زیر:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print(numbers[4::2])
```

خروجی:

```
[5, 7, 9]
```

نکته: برای خروج آنی از حلقه ها (for, while) می توانیم از دستور break استفاده کنیم.

مثال: با استفاده از دستور break می توانیم برنامه ی تشخیص اول بودن یا نبودن عدد را به صورت بهینه تری بنویسیم. در این حالت در حلقه ی for به محض آنکه مقسوم علیه ای برای عدد پیدا شود از حلقه بیرون رفته و اعداد دیگر را چک نخواهد کرد

```
x = int(input("addad ra vared kon: "))
b = False
for i in range(2, x):
    if x % i == 0:
        b = True
        break

if b is True:
    print("adad aval nist")
else:
    print("adad aval ast")
```

```
addad ra vared kon: 126
adad aval nist
```

خروجی:

نکته: دستور `continue` از تکرار دستوری که در حال اجراست می گذرد و به تکرار دستور بعدی می رود. در پایتون، زمانی که عبارت `continue` در داخل حلقه قرار می گیرد، تمام عبارات زیر آن را رد می کند و بلافاصله به تکرار بعدی پرش می کند.

نکته: در پایتون حلقه درون حلقه را به عنوان حلقه تو در تو می شناسیم. حلقه داخلی یا خارجی می تواند حلقه while یا حلقه for باشد.

مثال: برنامه ای بنویسید که جدول ضرب را تا 10*10 چاپ کند.

```python
for i in range(1, 11):
    for j in range(1, 11):
        print(f"{i} * {j} = {i * j}")
```

خروجی:

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
...
```

تمرین 1: برنامه ای بنویسید که تعداد ستاره های ردیف آخر را از کاربر گرفته و با توجه به آن اگر عدد زوج بود مانند شکل چپ و اگر فرد بود مانند شکل سمت راست.

```
**
****
******
********
```

```
*
***
*****
*******
```

پایان جلسه هشتم