هنر برنامه‌نویسی- لمس پایتون

مقدمات

گزارشی از محسن هوشمند و جواد اصغری

مقدمه

برنامه چیست: دنباله‌ای از دستوراتی که نحوهٔ اجرای رایانشی را مشخص می‌کند.

رایانش ممکن است مسئله‌ای ریاضی چون حل دستگاه معادلات یا ریشه چندجمله‌ای. همچنین می‌تواند رایانش حروف مانند جستجو و جانشانی متنی در مستندی یا شکلی گرافیکی چون پردازشتصویر یا پخش ویدئو باشد.

محیط‌های برنامه‌نویسی گوناگاگون دارای تفاوت هستند و امکانات متفاوتی را عرضه می‌کنند. اما همگی دارای امکاناتی بنیادی هستند که در زیر آنها را می‌بینیم:

- ورودی
- خروجی: نمایش روی صفحه نمایش، ذخیره در فایل، ارسال روی شبکه،جز اینها
- ریاضی: اجرای علمیات‌های ریاضی چون جمع و ضرب
- اجرای شرطی: وارسی شرط‌هی معین و اجرای کد مناسب برای هر یک
- تکرار: اجرای چندبارهٔ یک عمل، معمولا با تغییر یکی از متغیرها

اجزای کل زبان‌های برنامه‌نویسی. فارغ از پیچیدگی برنامه‌نویسی درنهایت اجزای هر زبان برنامه‌نویسی خارج از موارد بالا نیستند. برنامه‌نویسی را فانی شکستن کارهای پیچیده و بزرگ به زیروظایفی و وظایفی کوچنی است تا حدی که وظایف کوچک آنقدر ساده باشند که با یکی از دستورانت بنیادی اجرا شوند.

اجرای پیتون

نصب

امکان اجرای پیتون در مرورگر

یا اجرا روی سیستم شخصی

دو نسخه از پیتون: پیتون ۲ و پیتون ۳

پیتون ۳

پیتون زبانی مفسری است؟!

امکان تایپ python یا help() در نسخه فرمان جهت تشخیص نسخه و نوع آن

>>>

نمایشگر اینکه مفسر آماده ورودی کد. با نوشتن دستو ر و کلیک انتر (اجرا) یا در ژوپیتر ctrl+enter اجرای آن

با رسیدن به این مکان یعنی کاربر اماده کار در محیط پیتون و نوشتن کدهست. اکنون شما کدنویس شده‌اید و می‌توان امیدوار بود که رو به سوی برنامه‌نویس شدن دارید.

عملیات‌های ریاضی

```
>>>2+2
4
>>>2-2
0
```

```
>>>2*3
6
>>>4/2
2.0
```

تقسیم همیشه عدد اعشاری برمی‌گرداند

```
>>>5/2
2.5
>>>5%2
1
>>>5 / 2.0
2.5

>>>4.5 / 3
1.5
>>> 17 / 3 # برگرداندن عدد اعشاری
5.666666666666667

>>> 17 // 3 # برگرداندن بخش قبل ممیز
5
>>> 17 % 3 # علامت درصد برگرداندن باقی‌مانده
2
>>> 5 * 3 + 2 # بررسی درستی محاسبات باقی‌مانده و بخش غیراعشاری
17
>>>4**2
16
>>>4**2+2*2
20
```

مورد آخر ترکیب و اولویت را نشان می‌دهد. البته باید توجه داشت که پرانتز بیشترین اولویت را داراست. سوال 2**3-؟ چه مقداری را برمی‌گرداند؟

```
>>>4 ** 2 + 2*
File "<ipython-input-59-b57ad64e08d2>", line 1
    4 ** 2 + 2 *
                ^
SyntaxError: invalid syntax

>>> 9 ** ( 1 / 2)
3.0
```

```
>>> (50-5*6)/4
5.0
```

توجه کنید که اعدادی مانند 2، 4، 20،16 از نوع int یا عدد صحیح هستند، اما اعداد دارای بخش اعشاری مانند 5.0، 2.5، 1.5، 2.0 از نوع float یا اعاد اعشاری هستند. به سخن دیگر، پیتون همچون بسیاری دیگر از زبان‌های برنامه‌نیوسی درایا انواع متافوتی از نمایش داده است. در عملیات ترکیبی که شامل اعداد صحیح و اعشاری باشد

عملیات‌های بیتی

تقسیم صحیح

```
>>> 12 // 5
2

>>> 12 % 5
2

>>> 3 // 5
0

>>> -13 / 4
-3.25
>>> -13 // 4
-4

>>> 123 / 0
ZeroDivisionError                         Traceback (most recent call last
)
<ipython-input-53-76e1a9ab9410> in <module>
----> 1 123 / 0

ZeroDivisionError: division by zero
```

پیام بالا مشخص‌کنندهٔ موارد زیر است:

- استثنا یا خطای زمان اجرا
- مشخص کردن مکان خطا
- اعلام دستوری خطا
- نوع استثنا.

```
>>> 7.5 % 3.5
0.5
```

تقدم عملگرها و ترتیب عملیات

- پرانتز
- توان
- ضرب و تقسیم
- جمع و تفریق

عملیات‌های با اولویت یکسان از چپ به راست خوانده می‌شوند. این امر درباره توان استثناست. استفاده از پرانتزها توصیه می‌شود.

مقدار و نوع

2، 42.0 ، یا 'Hello, World!'

- مورد متقدم عدد صحیح یا integer
- مورد وسط عدد اعشاری یا floating point number
- مورد متاخر رشته یا string

```
>>> type(2)
int

>>> type(2.5)
float

>>> type('Tehran')
str
>>> type('2.5')
Str

>>> 2+-2
0
>>> 2++2
4
>>> 2+++2
4
>>> 2+*+2
File "<ipython-input-16-d1d7586f936c>", line 1
    2+*+2
      ^
SyntaxError: invalid syntax


>>> 2*+2
4

>>> 2+*2
4
```

چاپ و تابع چاپ

تایپ نوشته‌ای

>>> print('Hello, World!')

دستور چاپ ولی عدم چاپ روی کاغذ! بلکه نمایش در صفحه اجرا را موجب می‌شود. پاسخ با چیزی شبیه

Hello, World!

تفاوت دستور چاپ را در پیتون۲ و پیتون۳ دریابید.

تابع نام تابع پرانتز باز ورودی[ها] پرانتز بسته

زبان‌های صوری و طبیعی

زبان‌های طبیعی: زبانی که افراد و آحاد بدان صحبت کنند. زبان‌هایی مانند فارسی، عربی، ایتالیائی زبان‌های طبیعی هستند که افراد آنها را طراجی نکرده‌اند هر چند به دنبال اعمال دستوراتی و نظمی بر ان هستند. آنها طبیعی دچار تطورند.

زبان‌های صوری: زبان‌هایی که افراد برای کاربردهای خاص طراحی کرده‌اند. نمادگذاری ریاضی زبانی‌صوری است که جهت نمایش رابطه بین اعداد و علائم مناسب است. شیمی‌دانان از زبان صوری جهت نمایش ساهتاریهای شیمیایی مولکول‌ها استفاده می‌کنند.

به طریق اولی زبان‌های برنامه‌نمویسی نیز زبان‌هایی صوری هستند. که به منظور بیان و اظهار رایانش و محاسبه طراحی شده‌اند.

ذات زبان‌های صوری قواعد نحوی قوی و بی‌تغییری است که بر ساختار جملات و گزاره‌های فرامنتروایی کنند. مثلا، جمله یا گزارهٔ ۳ + ۳ = ۶ دارای نحوی درست است اما ۳+= ۶؛۳ نحوی درست ندارد. در شیمی نیز H_2O نحوا درست است، اما $ZZzZ_2$ گزاره‌ای نحوا درست نیست.

قواعد نحوی در دو سطح عریف می‌شوند: تکه‌ها و ساختار

تکه‌ها: اجزای بنیادی زبان چون کلمات، اعداد، و اجزای شیمیایی هستند. من باب مثال، علامت ؛ در ۳+= ۶؛۳ تکه‌ای معتبر نیست (یا حداقل در مثال مزبور چنین فرض می‌گیریم). به طریق مشابه، $ZZzZ_2$ نیز تکه‌ای معتبر در شیمی نیست.

رعایت «ساختار» درست حالت دوم نحو صحیح باشد. مثلا، ۳ +/۳ = ۶ ساختاری غلط دارد. هر چند که / و + هر دو تکه‌هائی درست باشند، اما نمی‌توان یکی را پس از دیگری به کار برد. یا در شمیی اندیس پس از اختصار عنصر می‌آید و نه قبل آن.

هنگام خواندن و قرائت جمله‌ای در زبان پارسی یا گزاره‌ای در زبان صوری نیاز است ساختار را بررسی کنید. امر مزبور در زبان طبیعی ناآگاهانه رخ می‌دهد. چنین فرایندی را تجزیه یا parsing خوانند.

تاکنون شباهت دو دسته زبان را گفتیم، اما تفاوت‌هایی نیز با یکدیگر دارند.

- زبان طبیعی دچار ابهام ambiguity است. زبان‌صوری تقریبا عاری از ابهام است.
- جهت کاهش ابهام زبان طبیعی سرشار از افزونگی است. در مقابل زبان‌های صوری افزونگی کمتر و ایجاز بیشتر دارند.
- استعارات: زبان طبیعی پر از ایهام و استعاره است. شخصی گوید «شتر دیدی ندیدی» نه شتری هست و نه شاهد شتری بلکه معنای آن اغماض و نادیده گرفتن است. در مقابل، زبان صوری گزاره دقیقا همان چیز را معنا می‌کند که بیان می‌کند.
- موسیقی کلمه: هم موسیقی کلمه و هم معنای آن. مثال شعر، وزن و آهنگ
- برنامه‌ها: برنامه رایانه‌ای بی‌ابهام و تحت‌اللفظی است و با تحلیل تکه‌ها و ساختار فهم‌پذیر است.

زبان‌های صوری فشردگی بیشتری نسبت به زبان‌های طبیعی دارند. در نتیجه فهم آنها زمان بیشتری لازم دارد. هرچند ساختار در فهم زبان صوری مهم است ولی خواندن از بالا به پایین یا چپ به راست در فهم چننی زبان‌هائی کارساز نیست یا کارایی خوبی نخواهد داشت. در عوض تجزیه برنامه در ذهن، مشخص کردن تکه‌ها و تفسیر ساختار کمک بیشتری خواهد کرد. اگر با شهرام درخشان در تبلیغی که در تلویزیون ایران عنوان می‌دارد «جزئیات نیز مهم است!» خطاهای کوچک در نگارش و نشانه‌گذاری بر عسکس زبان طبیعی موجب دردسرها و عدم اجرای درست در زبان‌های صوری خواهده شد.

خطایابی یا Debugging

اشتباه در برنامه‌نویسی لاجَرَم و لابد است. اصطلاح غریب حشره‌زدایی در انگلیسی به رفع آنها خطاب می‌شود. برنامه‌نویسی و خاصه خطایابی معمولا احساسات را دستخوش تحریک می‌کنند. دنباله رفع خطا شخص را عصبانی، افسرده، ناامید می‌کند. چون شخصی است که برنامه‌نویس را خوشحال، دمغ، خوابآلود،گیج، و جز اینها می‌سازد. پس حین برنامه‌نویسی لازم است تا آستین را بالا زد که «خطایابی وقت تعطیل نیست.» همان‌طور که دیگران گفته‌اند شاید راهی برای برنامه‌نیسی این باشد که رایانه را چون کارمندی زیردست دانست که نقاط قوتی درخشان و همچنین ضعف‌هایی در خود دارد. حال کار برنامه‌نویس این است که مدیری کارآمد باشد. مدیری که احساسشْ کارش و کارائی‌اش را ضایع نگردانَد.

خطایابی و رفع خطا آستانیْ هول‌انگیز است، اما در بهبود توانائی‌های برنامه‌نویسی امری اساسی است. پیگیری خطایابی شادی آورد و بوی بهبود و امید و اعتماد ارزانی کند.

اندیس‌ها

- حل مسئله، زبان‌های سطح بالا، زبان‌های سطح پائین، حمل‌پذیری، مفسر، پرامپت، برنامه، گزاره چاپ، عملگر، مقدار، نوع، عدد صحیح، عدد اعشاری، رشته، زبان طبیعی، زبان صوری، تکه، نحو، تجزیه، خطا و حشره، خطایابی و رفع آن

تمرین

- در دستور چاپ یکی یا هر دو پرانتزها را مگذارید. چه رخ می‌دهد؟
- اگر یکی یا هردو علامت نقل قول را قرار ندهید چه رخ می‌دهد؟
- ۲− + ۲ نتیجه‌اش چه می‌شود؟ ۲+ + ۲ نتیجه‌اش چه می‌شود؟ ۲+ + ۲؟ ۲+ *+ ۲؟ ۲ * + + ۲؟
- مفسر پیتون را به مثابهٔ ماشین حساب به کار گیرید و تعداد ثانیه‌ها ۴۲ ساعت و ۴۲ ثانیه را حساب کنید.
- هر ۱۰۰ کیلو چند من تبریز است.
- ۱۰ کیلومتر را در ۴۵ دقیق و ۵۴ ثانیه رفتید. سرعت متوسط شما چقدر است؟

متغیرها، عبارات، و گزاره‌ها

از ویژگی‌های زبان‌های برنامه‌نویسی که به آنها قدرت بالایی اعطا می‌کند استفاده و کار با متغیرها است. متغیرْ نامی است که به مقداری اشاره می‌کند.

گزاره‌های تخصیص: متغیری جدید را تعریف و مقداری را به آن اختصاص می‌دهد.

```
>>>message = 'khata'
>>>print(message)
Khata

>>>print('Khosh amadid!')

Khosh amadid!
```

چاپ و نمایش هر کلمه در یک خط جداگانه

```
>>>print('Khosh \namadid\n!')
Khosh
amadid
!
```

تب گذاری بین کلمات خروجی

```
>>>print('Khosh \tamadid\t!)
Khosh       amadid      !

>>>print('Khosh \\amadid!')
Khosh \amadid!

>>>print('Khosh \"amadid!')
Khosh "amadid!

>>>print('Khosh \'amadid!')
Khosh 'amadid!
```

عملیات‌های رشته

- عملگر + برای اتصال دو رشته
- عملگر * برای تکرار رشته

```
>>>n='salam '
>>>m ='alaik'

>>>print(n+'alaik')
salam alaik



>>>print(n*)
File "<ipython-input-77-e4d22623fa5a>", line 1
    print(n*)
           ^
SyntaxError: invalid syntax
```

```
>>>print(n*3)
salam salam salam
```

اما عمل اتصال متفاوت از عمل جمع اعداد است. موردی از تفاوت را شرح دهید.

```
>>>print(n * 3 + 2)
TypeError                                 Traceback (most recent call last
)
Input In [9], in <cell line: 1>()
----> 1 print(n * 3 + 2)

TypeError: can only concatenate str (not "int") to str

>>>a = 3

>>>print('alaik' + 3)
TypeError                                 Traceback (most recent call last
)
Input In [13], in <cell line: 1>()
----> 1 print('alaik' + 3)

TypeError: can only concatenate str (not "int") to str

>>>print(n + str(3))
salam 3

>>>print('Jame 7 + 3 = ', 7 + 3)
Jame 7 + 3 =  10


>>>print('Lotfan "Code" ra vared konid')
Lotfan "Code" ra vared konid


                                    >>>print('Lotfan 'Code' ra vared konid')
File "<ipython-input-81-4cdeef86ad07>", line 1
    print('Lotfan 'Code' ra vared konid')
                      ^
SyntaxError: invalid syntax


>>>print('Lotfan \'Code\' ra vared konid')
Lotfan 'Code' ra vared konid


>>>print("Hatman dar ebteda E'temad konid!")
Hatman dar ebteda E'temad konid!

>>>print("Lotfan \"Code\" ra vared konid")
Lotfan "Code" ra vared konid
```

توضیح سه نقل قولی

```
>>>print("""ba se "naqle qole" mokel 'hal' mishavad! """)
ba se "naqle qole" mokel 'hal' mishavad!
```

رشتهٔ چندخطی

```
>>>reshte_chand_khatti = """reshteye chand khati
... ran neveshtim"""
>>>print(reshte_chand_khatti)
reshteye chand khati
ran neveshtim


>>>reshte_chand_khatti
'reshteye chand khati\nran neveshtim'
```

گرفتن ورودی از کاربر

```
>>> Namm = input("Namat Chist? ")
Namat Chist? Mohsen

>>> Namm
'Mohsen'

>>> print(Namm)
Mohsen

>>> Naam = input("Namat Chist? ")
Namat Chist? Mohsen
```

تابع input همیشه رشته برمی‌گرداند

```
>>> mqdar1 = input('Vorode adade avval: ')
Vorode adade avval: 8

>>> mqdar2 = input('Vorode adade dovvom: ')
Vorode adade dovvom: 3

>>> mqdar1 + mqdar2
'83'
```

دریافت عدد صحیح از کاربر

```
>>> mqdar1 = input('Vorode adade avval: ')
Vorode adade avval: 8

>>> mqdar1 = int(mqdar1)
>>> mqdar1
8

>>> mqdar2 = input('Vorode adade digar: ')
Vorode adade digar: 3
```

```
>>> mqdar2 = int(mqdar2)
>>> mqdar2
3

>>> mqdar1 + mqdar2
11

>>> mqdar_qlat = int(input('Vorode adade: '))
Vorode adade: salam
ValueError                                Traceback (most recent call last
)
<ipython-input-117-2fe6d0c9f9ae> in <module>
----> 1 mqdar_qlat = int(input('Vorode adade: '))

ValueError: invalid literal for int() with base 10: 'salam'

>>> int('23')
23
>>> int(3.2)
3
>>> int(-3.22)
-3
```

نام‌های متغیر

```
>>> n = 17

>>> pi = 3.1415926535897932
```

امکان تعریف نام‌های بند. نام می تونا‌د ترکیبی از حروف و اعداد باشد. ولی منی‌توان با عدد شروع شود. امکان استفاده از جروف بزرگ و کوچک وجود دارد. ولی معمولا از حروف کوچک استفاده می‌شود.

زیرخط را می‌توان در نام استفاده کرد.

```
>>> 76trombones = 'big parade'
SyntaxError: invalid syntax
```

چرا؟ عدد در ابتدا

```
>>> more@ = 1000000
SyntaxError: invalid syntax
```

استفاده از علامت در

```
>>> class = 'Advanced Theoretical Zymurgy'
SyntaxError: invalid syntax
```

دلیل خطا؟ استفاده از کلیدواژه‌ها

کلیدواژه‌های پیتون: استفاده مفسر از آنها چهت تشخیص ساختار برنامه. از چنین کلماتی نمی‌توان برای نام متغیرها استفاده کرد.

کلیدواژه‌های پیتون ۳ شامل موارد زیر هستند:

False	class	finally	is	return	None	except
continue	for	lambda	try	True	def	in
from	nonlocal	while	and	del	global	raise
not	with	as	elif	if	or	
yield	assert	else	import	pass	break	

در مفسرهای معمولا کلیدواژه‌هی به رنگ و لون دیگر معمولا آبی نمایش داده می‌شوند و درنتیجه با تعریف ان به جای نام متغیر برنامه‌نویس متوجه خواهد شد.

عبارت: ترکیبی از مقادیر، متغیرها، و عملگرهاست. هر متغیر یا هر مقدار جداگانه نیز عبارت در نظر گرفته می‌شود.

فی‌المثل عبارات زیر را در نظر بگیرید:

```
>>> 42
42
>>> n
17
>>> n + 25
42
```

گزاره: قطعه کدی است که دارای اثر است. مثلا متغیری را ایجاد یا مقداری را نمایش می‌دهد

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

تخصیص مقدار ۱۷ به متغیر با نام n

```
>>> print(n)
```

گزاره چاپ که مقدار متغیر n را نمایش می‌دهد.

تصمیمات ساده و عملگرهای مقایسه

به عبرات دو مقدار (بولی) که مقدار صادق و کاذب (درست و نادرست) دریافت کند، شرط می‌گوئیم. دو نمونهٔ زیر را در نظر بگیرید.

```
>>> 7 > 4

True

>>> 7 < 4
False
```

از چند عملگر زیر جهت مقایسه شرطی استفاده می‌شود.

- > به معنای کوچکتر بودن سمت راست از سمت چپ
- < به معنای بزرگتر بودن سمت راست از سمت چپ

- => به معنای کوچکتر بودن سمت راست یا برابر بودن دو مقدار
- =< به معنای بزرگتر بودن سمت راست یا برابر بودن دو مقدار
- == به معنای برابر بودن دو مقدار
- = ! به معنای نابرابر بودن دو مقدار

```
7 > = 4
File "<ipython-input-122-5c6e2897f3b3>", line 1
    7 > = 4
        ^
SyntaxError: invalid syntax
```

توضیحات

با بزرگتر شدن و پیچیده‌تر شدن برنامه،خواند و مدیریت آن نیز مشکل‌تر می‌شود. همان‌گونه که بالاتر ذکر آن رفت، زبان‌های صوری موجز هستند و فراموشی معنای متناظر آنها سخت آسان رخ می‌دهد. مثلا اگر کدی را دید نمی‌توان به سرعت دریافت چه می‌کند یا چرا از آن استفاده می‌شود. بنابراین افزودن توضیحات راهگشاست. توضیحات قطعه متن‌هایی هستند که در متن کد می‌ایند ولی در احرا تاثیری ندارند و صرفا با هداف اطلاع خواننده کاربرد دارند. علامت آغاز توضیح در پیتون # است.

```
>>># mohasebeye nesbat bar asase sa’at
>>>percentage = (minute * 100) / 60
```

یا

```
>>>percentage = (minute * 100) / 60 # nesbat bar asase sa’at
```

به سخن دیگر هر چیز پس از علامت # جزو بخش اجرایی کد به حساب نمی‌آید.

توضیحات بی‌فایده و فایده‌مند

```
>>>v = 5 # Takhsis 5 be v
```

```
>>>v = 5 # Sorat ba meter/second.
```

به سخن دیگر، مفسر آنها را نادیده می‌گیرد و در اجرای نهایی تاثیری ندارند. توضیحات جهت واضح کردنم کد و همچنین مستندسازی برنامه به کار می‌رود.

تصمیم‌گیری با دستور if

```
>>>"""Moqayeseye do adad ba "if" o maalgarhaye moqayese"""

>>>print("Lotfan do adad jahat moqayese vared konid.")

>>># Khandan adad Avval
>>>adad1 = int(input("Adad nakhost ra vared konid: "))
```

```
>>># Khandan adad Dovvom
>>>adad2 = int(input("Adad dovvom ra vared konid: "))

>>>if adad1 == adad2:
>>>    print(adad1, ' barabare ba ', adad2)

>>>if adad1 != adad2:
>>>    print(adad1, ' namosavi ba ', adad2)

>>>if adad1 < adad2:
>>>    print(adad1, ' Kochekltar az ', adad2)

>>>if adad1 > adad2:
>>>    print(adad1, ' Bozorgatar az ', adad2)

>>>if adad1 <= adad2:
>>>    print(adad1, ' Mosavi  ba ya Kochekltar az ', adad2)

>>>if adad1 >= adad2:
>>>    print(adad1, ' Mosavi  ba ya Bozorgatar az ', adad2)


Lotfan do adad jahat moqayese vared konid.
Adad nakhost ra vared konid: 8
Adad dovvom ra vared konid: 4
8  namosavi ba  4
8  Bozorgatar az  4
8  Mosavi  ba ya Bozorgatar az  4
```

مد متنی (اسکرپیت)

تاکنون در مدل تعاملی که به نوعی برنامه‌نویس در تعامل مستقیم با مفسر است. مد تعاملی شاید برای آغاز مناسب باشد ولی برای کار درست و حسابی کارا نیست.

روش دیگر استفاده از فایل جهت ذخیرهٔ کد است که به اسکریپت شهره است. حال می‌توان مفسر را در حال اسکریپتی روی متن کد اجرا کرد. قرارداد شده است که اسکریپت‌های پیتونی داریا پسوند *py*. هستند. مثلا، برنامه بالا را در فایل با فرمت *py* ذخیره می‌کنیم.

```python
"""Moqayeseye do adad ba "if" o amalgarhaye moqayese"""

print("Lotfan do adad jahat moqayese vared konid.")

# Khandan adad Avval
adad1 = int(input("Adad nakhost ra vared konid: "))

# Khandan adad Dovvom
adad2 = int(input("Adad dovvom ra vared konid: "))

if adad1 == adad2:
    print(adad1, ' barabare ba ', adad2)

if adad1 != adad2:
    print(adad1, ' namosavi ba ', adad2)

if adad1 < adad2:
    print(adad1, ' Kochekltar az ', adad2)

if adad1 > adad2:
    print(adad1, ' Bozorgatar az ', adad2)

if adad1 <= adad2:
    print(adad1, ' Mosavi  ba ya Kochekltar az ', adad2)

if adad1 >= adad2:
    print(adad1, ' Mosavi  ba ya Bozorgatar az ', adad2)
```

حال آن را با انواع برنامه‌های می‌توان اجرا کرد. اما در اینجا لازم است خطوطدستورات را توضیحی دهیم. خطوط ۱، ۵، و ۷ از نوع توضیح هستند. خط اول در واقع به نوع راهنمایی برای برنامه است که هدف کلی برنامه را توضیح می‌دهد. بعضی خطوط مانند خطوط ۲، ۴، ۷، و چند خط دیگر خطوط خالی هستند و در اجرای کد تاثیری ندارند. باز جهت خوانایی معمولا فواصلی لحاظ می‌شوند. خطوط ۶ و ۹ دو عدد صحیح را از کاربر با کمک دستورات *int* و *input* دریاف می‌کند. دستورات *if* جهت مقایسهٔ دو عدد صحیح ورودی استفاده می‌شود. دستور *if* دارای چهار بخش است.

- کلیدواژهٔ *if*
- شرط و وضعیتی که باید ارزیابی شود.
- دونقطه :
- یک یا چند دستور که در خط بعد و با تورفتگی یا دندانه می‌آیند و در صورت برقراری شرط اجرا می‌شوند.

دونقطه در انتهای خط اول الزامی و مهم است و فراموشی آن از خطاهای رایج است.

همچنین تورفتگی دستورات بسیار مهم است. تورفتگی پیمانه‌بندی را مشخص می‌کنند. معمولا تورفتگی بر اسا راهنمای استاندارد پایتون چهار حرف فاصله است.

مورد مهم دیگر مقایسه == و = است مورد نخست به تساوی می‌پردزا و زا نوع مقایسه است ولی مورد دو جهت انتساب بکار می‌رود. در بررسی‌های و مقایسه‌های منطقی دو مورد حتما باید از مود نخست استفاده کرد و از مورد اخیر اجتناب کرد.

عملگرهای مقایسه را می‌توان ترکیب کرد تا فرضا حضور مقداری را در بازه‌یا وارسی کرد. فرضا بررسی می‌کنیم ورودی در بازه $[0,10]$ قرار دارد یا نه.

```
>>>x = 4

>>>1 <= x <= 10
True

>>>y = 11

>>>1 <= y <= 10
False
```

عملگرهای مقایسه نیز دارای تقدم و تأخرند. بنابراین جدول کاملتر تقدم‌ها را به صورت زیر بازنویسی می‌کنیم.

عملگر	ترتیب اجرا
()	چپ به راست
**	**راست** به چپ
* / // %	چپ به راست
+ -	چپ به راست
> >= < <=	چپ به راست
== !=	چپ به راست
=	**راست** به چپ

مقدماتی تکمیلی دربارهٔ متغیرها

در پایتون عدد صحیح، عدد اعشاری، و رشته ها همگی شی به حساب می‌آیند. هر شی نیز دارای نوع و مقدار است. همان‌طور که قبلا ذکر آن رفت، نوع با دستور *type* مشخص می‌شود.

همچنین هر متغیر به شی‌ای ارجاع می‌دهد. پس و از ایاد پیوند،می‌توان از متغیر برای دست یافتن به شی استفاده کرد.

```
>>>x = 7


>>>x + 10
17


>>>x
7
```

اما برای تغییر دادن آن می‌توان به صورت زیر عمل کرد.

```
>>>x = x + 10

>>>x
17
```

در پایتون می‌توان نوع متغیر را عوض کرد. مثال زیر را ببینید.

```
>>>type(x)
```

```
int
>>>x = 3.5
>>>type(x)
float
>>>x = 'Tehran'
>>>type(x)
str
```

با ایجاد شی پایتون مکانی را در حافظه به آن تخصیص می‌دهد. و در صورت عدم نیاز آ» را از حافظه پاک می‌کند. پایتون شی بدون ارجاع را به صورت خودکار از حافظه پاک می‌کند که به فرایند مذکور بازیافت بازیافت حافظه گفته می‌شود.

مثالی آماری

در علم داده از آمار جهت توصیف و خلاصه‌سازی داده‌ها استفاده می‌شود. از موارد مهم آما توصیفی شامل کمینه، بیش، دامه، شمارش، و مجموعه است. پس چند آمار توصیفی را در ادامه بررسی می‌کنیم.

مثال پائین سه عدد دریافت می‌کند و کمینه آنها را حساب می‌کند.

```
"""Yaftan kamineye se adad"""
# Khandan adad Avval
adad1 = int(input("Adad nakhost ra vared konid: "))

# Khandan adad Dovvom
adad2 = int(input("Adad dovvom ra vared konid: "))

# Khandan adad Sevvom
adad3 = int(input("Adad sevvom ra vared konid: "))

kam = adad1

if adad2 < kam:
    kam = adad2
if adad3 < kam:
    kam = adad3

print('Mqdar kam: ', kam)
```

Adad nakhost ra vared konid: 2

Adad dovvom ra vared konid: 4

Adad sevvom ra vared konid: -12

Mqdar kam: -12

یافتن مقدار بیشینهٔ سه عدد

```python
"""Yaftan bishineye se adad"""
# Khandan adad Avval
adad1 = int(input("Adad nakhost ra vared konid: "))

# Khandan adad Dovvom
adad2 = int(input("Adad dovvom ra vared konid: "))

# Khandan adad Sevvom
adad3 = int(input("Adad sevvom ra vared konid: "))

bish = adad1

if adad2 > bish:
    bish = adad2
if adad3 > bish:
    bish = adad3

print('Mqdar bish: ', bish)
```

```
Adad nakhost ra vared konid: 2
Adad dovvom ra vared konid: 4
Adad sevvom ra vared konid: 3
Mqdar bish:  4
```

تمرین-

- جمع سه عدد ورودی را بنویسید.
- در مثال‌های اسکریپتی اگر ورودی عدد صحیح نباشد چه اتفاقی می‌افتد؟ راه‌حل چیست؟
- در مثال یافتن کمینه اگر دو مقدار از برابر یکدیگر و کمتر از مقدار سوم باشند، برنامه کدامیک را برمی‌گرداند؟
- کدامیک از انتساب‌ها درست است؟ n = 42؟ ;x = y = 1؟ xy به جای x*y؟
- حجم کره با شعاع r برابر است با $\frac{۴}{۳}\pi r^{۳}$. حجم کره‌ای با شعاع ۵ چقدر است؟

خطایابی

در برنامه انواع خطاها امکان اتفاق دارند. می‌توان آنها را در سه دسته معرفی کرد: خطاهای نحوی، خطالهای زمان احرا، خطاهای معنائی.

- خطاء نحوی *Syntax error*
  - به خطاهای دستوری و ساختاری برمی‌گردد. با وقوع خطای نحوی کد اجرا نخواهد شد و نیاز است آن را رفع کرد. با کسب تجربهٔ بیشتر خطاهای نحوی کمتر و کمتر رخ خواهند داد.

- خطاء زمان‌اجرا *Runtime error* یا *exceptions*
  - خطا خود را در زمان اجرا آشکار خواهد کرد. در بخش‌های بعدی و کدهای پیچیده بیشتر دیده خواهد شد.
- خطای معنائی *Semantic error*
  - خطاهای مزبور به معنا ارتباط می‌یابند. در واقع اجرای چنین خطاهایی پیغام خطایی را مسبب نمی‌شود، ولی کار درست را انجام نمی‌دهد و از منطق و دلیل اجرای آن عدول خواهد کرد. یافتن چنین خطاهایی معمولا سخت است.

اندیس:

منابع

A. Downey, Think Python2, How to Think Like a Computer Scientist, Green Tea Press, 2nd ed., 2015.

P. J. Deitel, H. Deitel, “Introduction to Python for Computer Science and Data Science: Learning to Program with AI, Big Data and the Cloud,” Pearson, 2021.