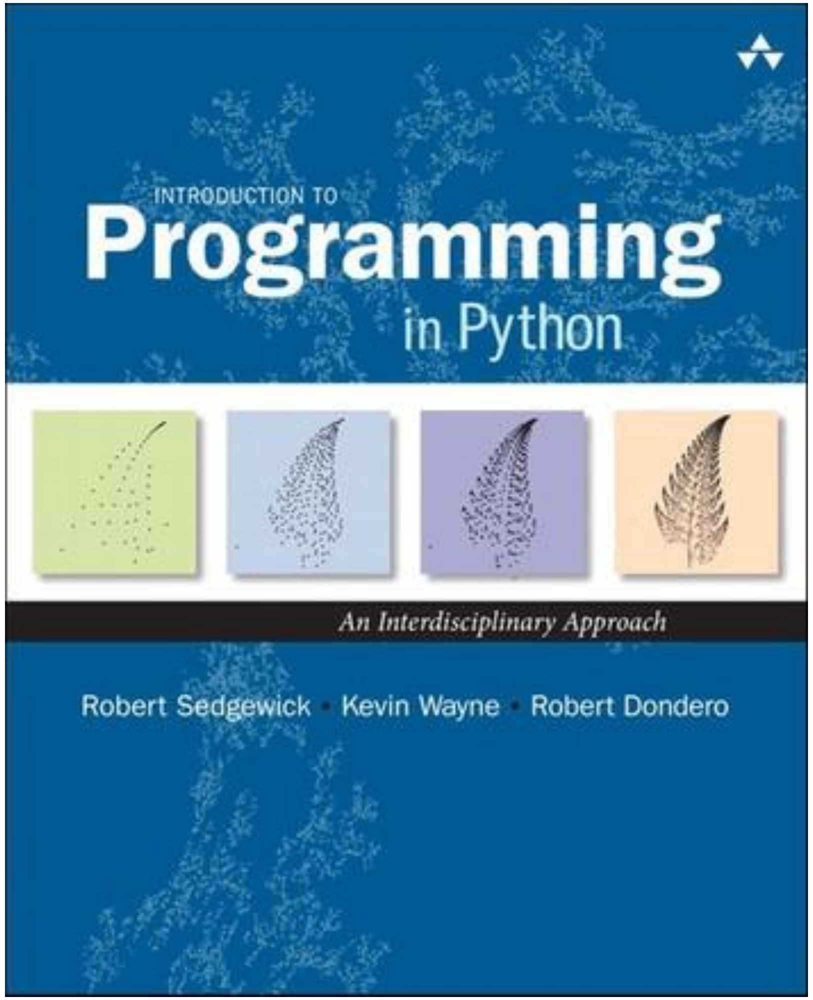
عناصر برنامه‌نویسی: جریان کنترل

سید ناصر رضوی **www.snrazavi.ir**
۱۳۹۸

۳-۱ دستورات شرطی و حلقه‌ها



INTRODUCTION TO
Programming
in Python

An Interdisciplinary Approach

Robert Sedgewick · Kevin Wayne · Robert Dondero

اجزای برنامه‌نویسی

any program you might want to write

objects

functions and modules

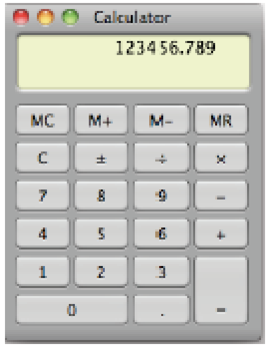
graphics, sound, and image I/O

arrays

conditionals and loops

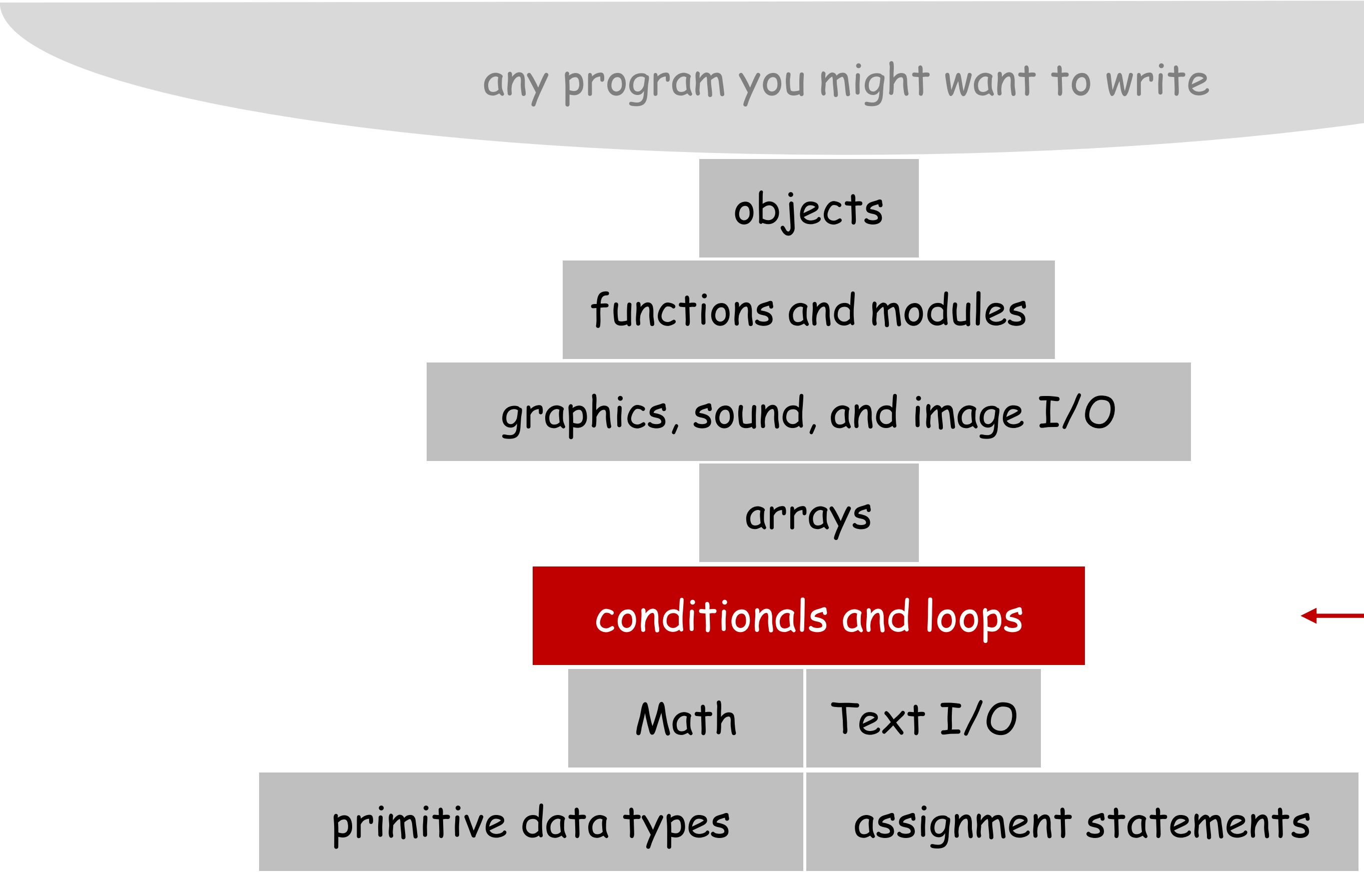
Math | Text I/O

primitive data types | assignment statements



تا اینجا:
کامپیوتر به عنوان
یک ماشین حساب

اجزای برنامه‌نویسی



any program you might want to write

objects

functions and modules

graphics, sound, and image I/O

arrays

conditionals and loops

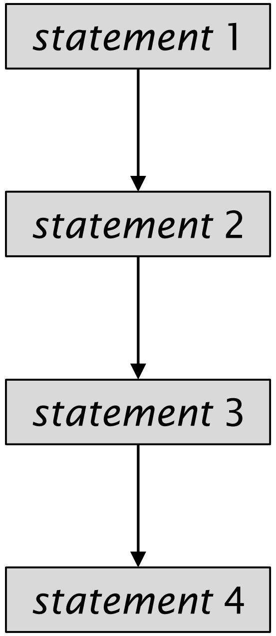
Math | Text I/O

primitive data types | assignment statements

به سوی بی‌نهایت!

جریان کنترل

- جریان کنترل.
  - دنباله‌ای از دستورات در برنامه که واقعاً اجرا می‌شوند.
  - دستورات شرطی و حلقه‌ها: تغییر جریان کنترل.



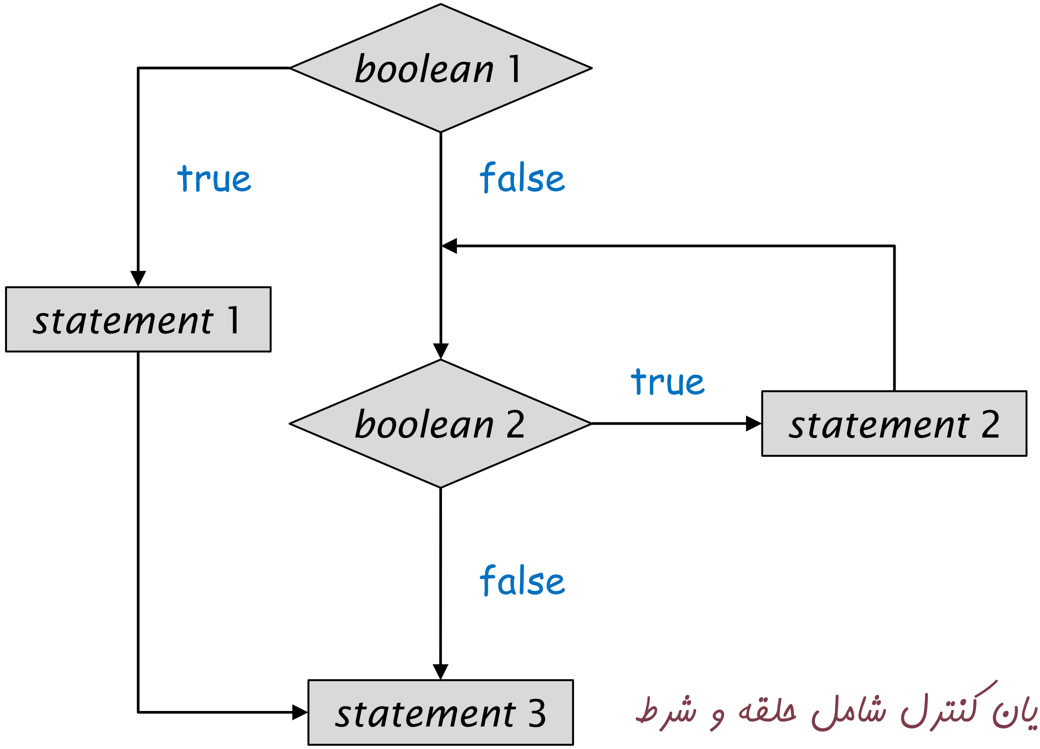
جریان کنترل خط مستقیم

دستورات یکی پس از دیگری به ترتیب مشخص شده اجرا می‌شوند

جریان کنترل

- جریان کنترل.
  - دنباله‌ای از دستورات در برنامه که واقعاً اجرا می‌شوند.
  - دستورات شرطی و حلقه‌ها: تغییر جریان کنترل.

بسیاری از برنامه‌ها به جریان کنترل پیچیده‌تری نیاز دارند

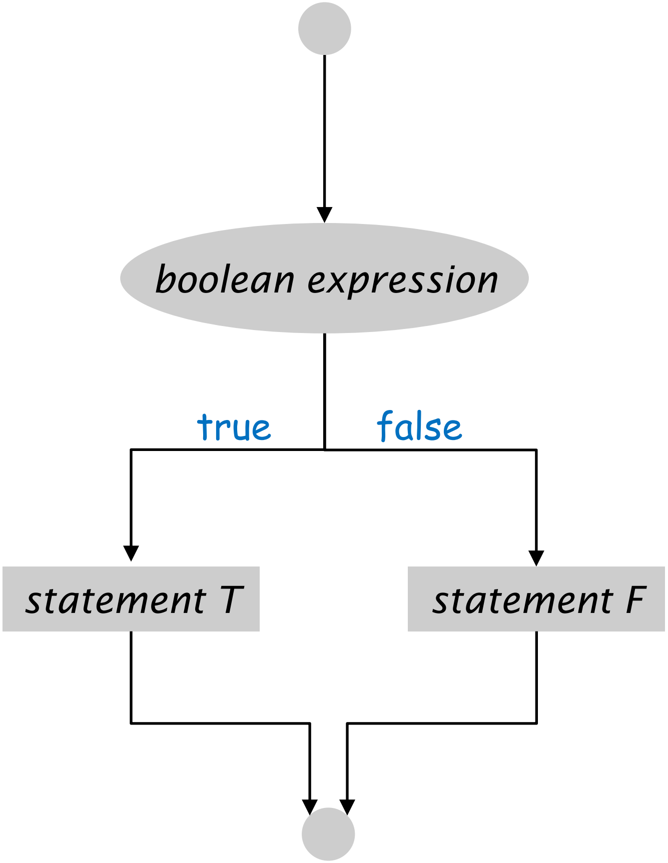


جریان کنترل شامل حلقه و شرط

دستورات شرطی

دستور if

- دستور `if`. یک ساختار انشعاب متداول.
  - یک عبارت بولی ارزیابی می‌شود.
  - اگر حاصل `true` بود، آنگاه یک مجموعه از دستورات اجرا می‌شوند.
  - اگر حاصل `false` بود، آنگاه یک مجموعه دیگر از دستورات اجرا می‌شوند.



```
if boolean expression:
    statement T
else:
    statement F
```

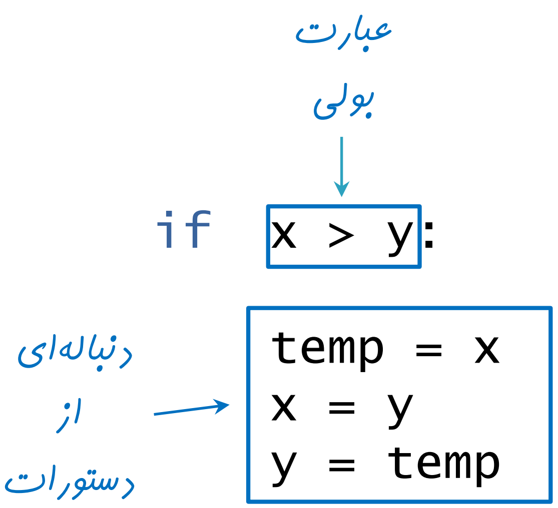
در اینجا هر مجموعه از دستورات می‌تواند قرار بگیرد.

دستور if

- ☐ دستور `if`. یک ساختار انشعاب متداول.
  - یک عبارت بولی ارزیابی می‌شود.
  - اگر حاصل `true` بود، آنگاه یک مجموعه از دستورات اجرا می‌شوند.
  - اگر حاصل `false` بود، آنگاه یک مجموعه دیگر از دستورات اجرا می‌شوند.

```
if  x > y:
    maximum = x
else:
    maximum = y
```

```
if  x < 0:
    x = -x
```



عبارت بولی

دنباله‌ای از دستورات

```
if  x > y:
    temp = x
    x = y
    y = temp
```

```
if  x > y:
    x, y = y, x
```

دستور if

- مثال. شبیه‌سازی پرتاب یک سکه.

```
import random


if random.randrange(0, 2) == 0:
    print(‘Heads‘)
else:
     print(‘Tails‘)
```

```
% python flip.py
Heads

% python flip.py
Heads

% python flip.py
Tails

% python flip.py
Heads
```

چند مثال از کاربرد دستور if

absolute value

```
if x < 0:
   x = -x
```

put x and y into sorted order

```
if x > y:
   temp = x
   x = y
   y = temp
```

maximum of x and y

```
if x > y: maximum = x
else:     maximum = y
```

error check for remainder operation

```
if den == 0: print(‘Division by zero’)
else:        print(‘Remainder = ” + num % den)
```

error check for quadratic formula

```
discriminant = b*b - 4.0*a*c
if discriminant < 0:
    print(‘No real roots’)
else:
    d = math.sqrt(discriminant)
    print((-b + d) / 2.0)
    print((-b - d) / 2.0)
```

اهمیت تورفتگی (دندانه‌گذاری)

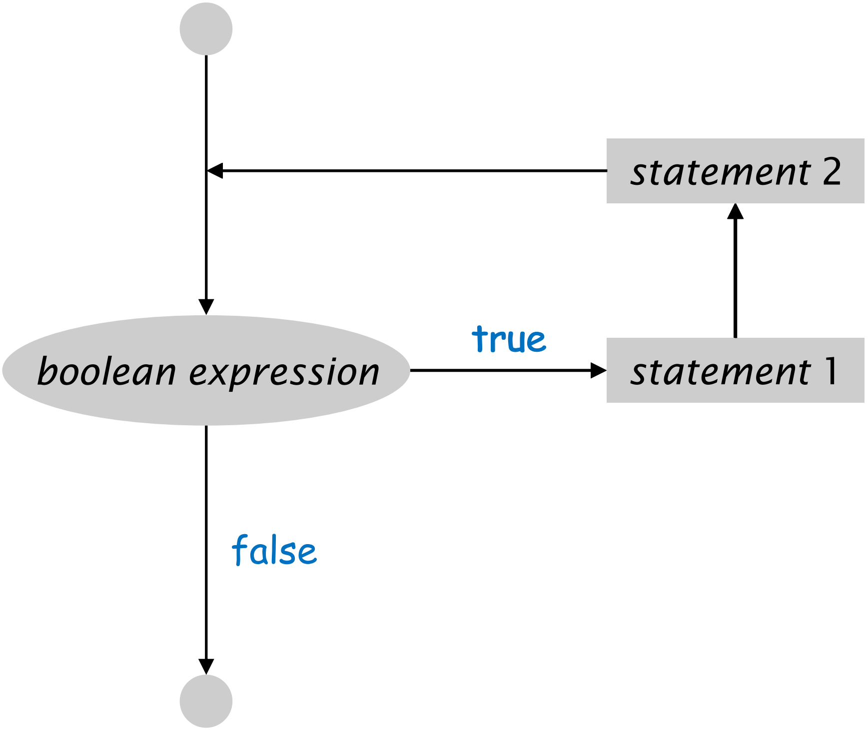
```
if x >= 0:
    print(‘Not ‘, end=‘’)
print(‘negative‘)
```

```
if x >= 0:
    print(‘Not ‘, end=‘’)
    print(‘negative‘)
```

حلقه while

حلقه while

- حلقه `while` یک ساختار تکرار متداول.
  - یک عبارت بولی ارزیابی می‌شود.
  - اگر حاصل `true` بود، تعدادی دستور اجرا می‌شود.
  - تکرار.



```
while  boolean expression:
    statement 1
    statement 2
```

بدنه حلقه

مثال: حلقه while

```
print(‘1st Hello’)
print(‘2nd Hello‘)
print(‘3rd Hello‘)

i = 4
while i <= 10:
    print(str(i) + ‘th Hello‘)
    i = i + 1
```

```
% python tenhellos.py

1st Hello
2nd Hello
3rd Hello
4th Hello
5th Hello
6th Hello
7th Hello
8th Hello
9th Hello
10th Hello
```

یک خطای متداول: فراموش کردن دستور افزایش شمارنده حلقه

مثال: چاپ توان‌های عدد ۲

```
import sys

n = int(sys.argv[1])

i = 0
power = 1
while i <= n:
    print(str(i) + ‘ ‘ + str(power))
    i = i + 1
    power = 2 * power
```

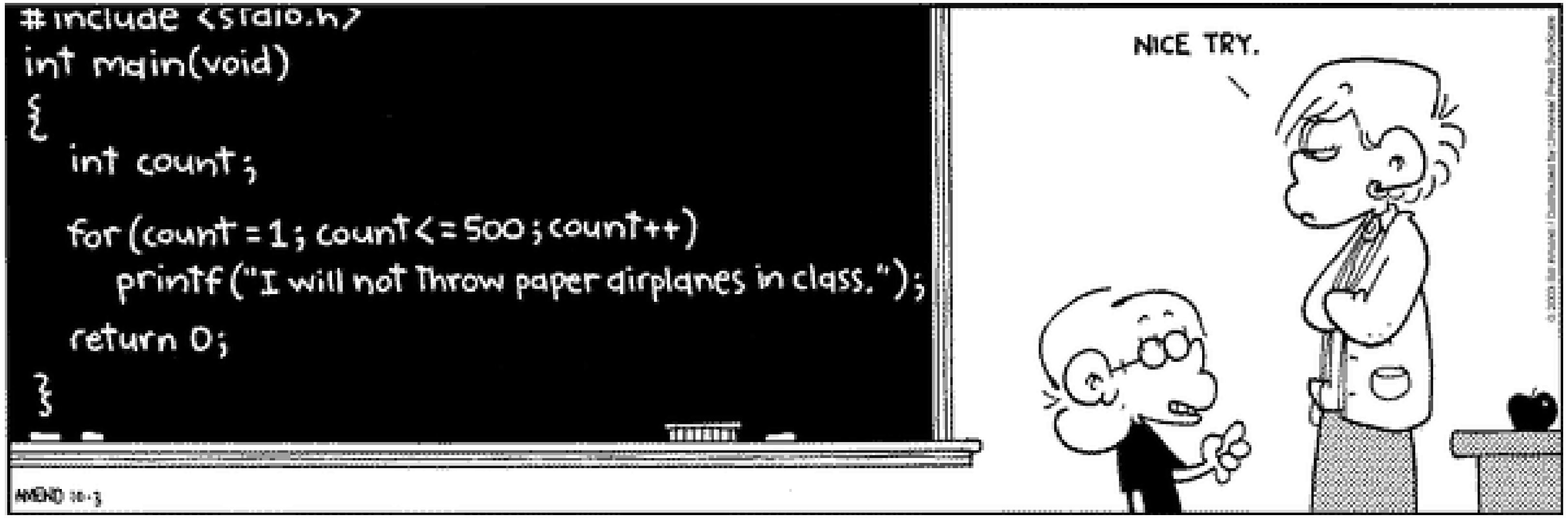
```
% python powersoftwo.py 0
0 1

% python powersoftwo.py 1
0 1
1 2

% python powersoftwo.py 2
0 1
1 2
2 4
```

مثال: چاپ توان‌های عدد ۲

```python
import sys

n = int(sys.argv[1])

i = 0
power = 1
while i <= n:
    print(str(i) + ‘ ‘ + str(power))
    i += 1        # i = i + 1
    power *= 2    # power = 2 * power
```

```
% python powersoftwo.py 0
0 1

% python powersoftwo.py 1
0 1
1 2

% python powersoftwo.py 2
0 1
1 2
2 4
```

حلقه for



```
#include <stdio.h>
int main(void)
{
  int count;

  for (count = 1; count <= 500; count++)
     printf("I will not Throw paper airplanes in class.");
  return 0;
}
```

مثال: حلقه for

حلقه for

```
print(‘1st Hello’)
print(‘2nd Hello‘)
print(‘3rd Hello‘)


for i in range(4, 11):
    print(str(i) + ‘th Hello‘)
```

حلقه while

```
print(‘1st Hello’)
print(‘2nd Hello‘)
print(‘3rd Hello‘)

i = 4
while i <= 10:
    print(str(i) + ‘th Hello‘)
    i = i + 1
```

مثال: حلقه for

حلقه for

```
print(‘1st Hello’)
print(‘2nd Hello‘)
print(‘3rd Hello‘)



for i in range(4, 11):
    print(str(i) + ‘th Hello‘)
```

```
range(m, n)    →    m <= i < n
```

```
range(n)    →    0 <= i < n
```

مثال: چاپ توان‌های عدد ۲

```
import sys

n = int(sys.argv[1])


power = 1
for i in range(n + 1):
    print(str(i) + ‘ ‘ + str(power))
    power *= 2
```

حلقه for

```
import sys

n = int(sys.argv[1])

i = 0
power = 1
while i <= n:
    print(str(i) + ‘ ‘ + str(power))
    i += 1
    power *= 2
```

حلقه while

ترسیم خطکش

- ایجاد بخش‌بندی‌های مربوط به یک خطکش.
  - مقدار رشته `ruler` را برابر با ‘ ‘ قرار بده.
  - به ازای هر مقدار `i` از 1 تا n:

    دو نسخه از مقدار رشته `ruler` را در دو طرف مقدار `i` قرار بده.

```
import sys

n = int(sys.argv[1])

ruler = ‘ ‘
for i in range(1, n + 1):
    ruler = ruler + str(i) + ruler
print(ruler)
```

i	ruler
	" "
1	" **1** "
2	" 1 **2** 1 "
3	" 1 2 1 **3** 1 2 1 "
4	" 1 2 1 3 1 2 1 **4** 1 2 1 3 1 2 1 "

ترسیم خطکش

- مشاهده. حلقه‌ها می‌توانند حجم عظیمی از خروجی را تولید کنند.

```
% python ruler_n.py 1
 1

% python ruler_n.py 2
 1 2 1

% python ruler_n.py 3
 1 2 1 3 1 2 1

% python ruler_n.py 4
 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

% python ruler_n.py 5
 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 5 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

% python ruler_n.py 100
 MemoryError
```

چند مثال از حلقه‌ها

write first n + 1 powers of 2

```
power = 1
for i in range(n + 1):
    print(str(i) + ‘ ‘ + str(power))
    power *= 2
```

compute a finite sum
(1 + 2 + … + n)

```
total = 0
for i in range(1, n + 1):
    total += i
print(total)
```

compute a finite product
(n! = 1 * 2 * … * n)

```
product = 1
for i in range(1, n + 1):
    product *= i
print(product);
```

print a table of function values

```
for i in range(n + 1):
    print(str(i) + ‘ ‘ + str(2 * math.pi * i / n))
```

ساختارهای تو در تو

دستورات if تو در تو

- ☐ مثال. پرداخت یک نرخ مالیات خاص بر مبنای سطح درآمد.

```
if income < 0:
    rate = 0.00
else:
    if income < 8925:
        rate = 0.10
    else:
        if income < 36250:
            rate = 0.15
        else:
            if income < 87850:
                rate = 0.23
            ...
```

Income	Rate
< 0	0 %
0 - 8,925	10%
8,925 – 36,250	15%
36,250 – 87,850	23%
87,850 - 183,250	28%
183,250 - 398,350	33%
398,350 - 400,000	35%
400,000 -	39.6%

دستورات if تو در تو

- ☐ مثال. پرداخت یک نرخ مالیات خاص بر مبنای سطح درآمد.

```
if   income < 0:      rate = 0.00
elif income < 8925:   rate = 0.10
elif income < 36250:  rate = 0.15
elif income < 87850:  rate = 0.23
elif income < 183250: rate = 0.28
elif income < 398350: rate = 0.33
elif income < 400000: rate = 0.35
else:                 rate = 0.396
```

Income	Rate
< 0	0 %
0 - 8,925	10%
8,925 - 36,250	15%
36,250 - 87,850	23%
87,850 - 183,250	28%
183,250 - 398,350	33%
398,350 - 400,000	35%
400,000 -	39.6%

دستورات if تو در تو: چالش

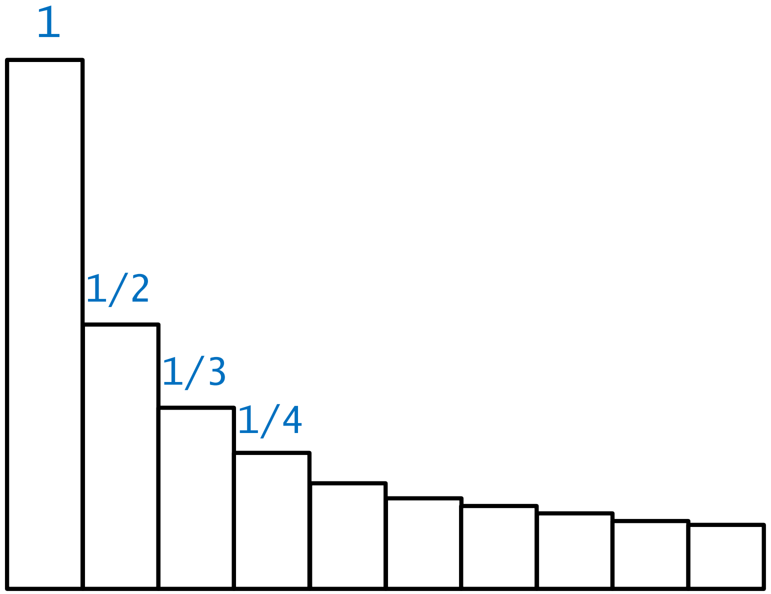
- **پرسش.** در محاسبات زیر برای محاسبه نرخ مالیات، چه چیزی نادرست است؟

```
if income < 0:      rate = 0.00
if income < 8925:   rate = 0.10
if income < 36250:  rate = 0.15
if income < 87850:  rate = 0.23
if income < 183250: rate = 0.28
if income < 398350: rate = 0.33
if income < 400000: rate = 0.35
else:               rate = 0.396
```

Income	Rate
< 0	0 %
0 - 8,925	10%
8,925 - 36,250	15%
36,250 - 87,850	23%
87,850 - 183,250	28%
183,250 - 398,350	33%
398,350 - 400,000	35%
400,000 -	39.6%

چند کاربرد: سری هارمونی

```
import sys

n = int(sys.argv[1])

total = 0
for i in range(1, n + 1):
    total += 1.0 / i

print(total)
```



چند کاربرد: سری هارمونی

```python
import sys

n = int(sys.argv[1])

total = 0
for i in range(1, n + 1):
    total += 1.0 / i

print(total)
```

```
% python harmonic.py 2
1.5

% python harmonic.py 10
2.9289682539682538

% python harmonic.py 10000
9.787606036044348
```

محاسبه جذر اعداد

- ☐ هدف. پیاده‌سازی تابع `math.sqrt()`
- ☐ روش نیوتن-رافسون به منظور محاسبه ریشه دوم عدد c
  - ▫ مقدار t_0 را برابر با c قرار بده.
  - ▫ تا زمانی که شرط $t_i = c/t_i$ برقرار نشده، عمل زیر را تکرار کن:
    مقدار t_{i+1} را برابر با میانگین مقادیر t_i و c/t_i قرار بده.

```
% python sqrt.py 2.0
1.414213562373095
```

$$
\begin{aligned}
t_0 & & &= 2.0 \\
t_1 &= \tfrac{1}{2}\left(t_0+\tfrac{2.0}{t_0}\right) &&= 1.5 \\
t_2 &= \tfrac{1}{2}\left(t_1+\tfrac{2.0}{t_1}\right) &&= 1.4166666666666665 \\
t_3 &= \tfrac{1}{2}\left(t_2+\tfrac{2.0}{t_2}\right) &&= 1.4142156862745097 \\
t_4 &= \tfrac{1}{2}\left(t_3+\tfrac{2.0}{t_3}\right) &&= 1.4142135623746899 \\
t_5 &= \tfrac{1}{2}\left(t_4+\tfrac{2.0}{t_4}\right) &&= 1.414213562373095
\end{aligned}
$$

محاسبه جذر عدد ۲

محاسبه جذر اعداد

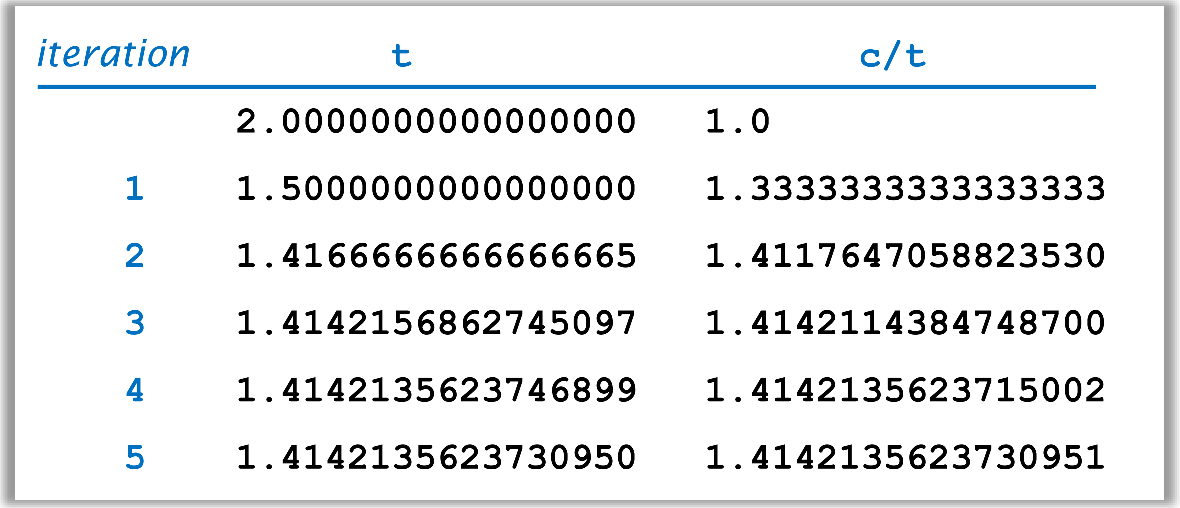
- هدف. پیاده‌سازی تابع `math.sqrt()`
- روش نیوتن-رافسون به منظور محاسبه ریشه دوم عدد c
  - مقدار t_0 را برابر با c قرار بده.
  - تا زمانی که شرط $t_i = c/t_i$ برقرار نشده، عمل زیر را تکرار کن:
    مقدار t_{i+1} را برابر با میانگین مقادیر t_i و c/t_i قرار بده.

```python
import sys

EPSILON = 1e-15

c = float(sys.argv[1])
t = c
while abs(t - c/t) > (EPSILON * t):
    t = (c/t + t) / 2.0

print(t)
```

```
% python sqrt.py 2.0
1.414213562373095
```

محاسبه جذر اعداد: ردیابی اجرا

```
import sys

EPSILON = 1e-15

c = float(sys.argv[1])
t = c
while abs(t - c/t) > (EPSILON * t):
    t = (c/t + t) / 2.0

print(t)
```

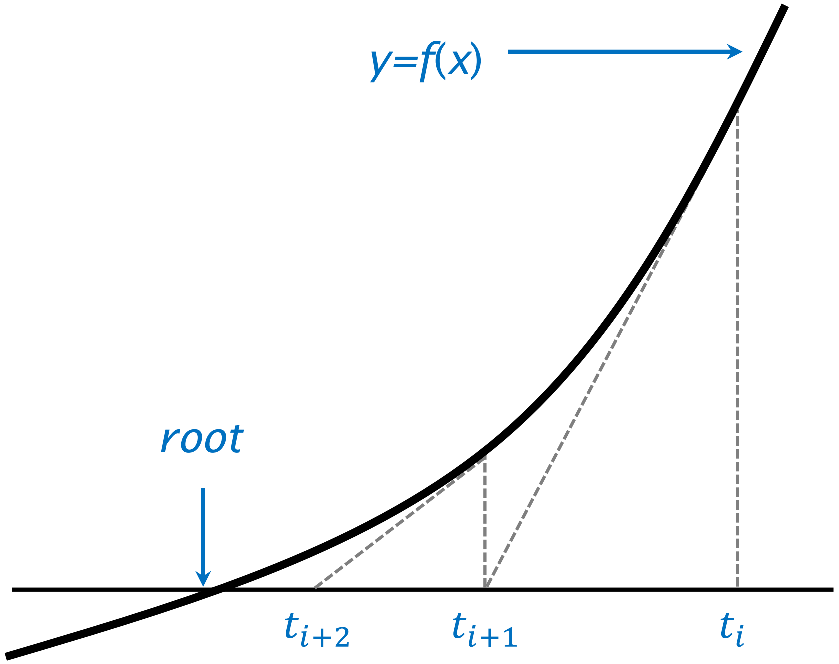


iteration	t	c/t
	2.0000000000000000	1.0
1	1.5000000000000000	1.3333333333333333
2	1.4166666666666665	1.4117647058823530
3	1.4142156862745097	1.4142114384748700
4	1.4142135623746899	1.4142135623715002
5	1.4142135623730950	1.4142135623730951

روش نیوتن-رافسون

- روش محاسبه ریشه دوم.
  - هدف: یافتن ریشه هر تابع مانند f(x)
  - با تخمین t_0 شروع کن.
  - خط مماس بر منحنی را در $x = t_i$ ترسیم کن.
  - مقدار t_{i+1} را برابر با مختصات x نقطه برخورد خط مماس و محور x قرار بده.
  - مراحل فوق را تا رسیدن به دقت دلخواه تکرار کن.

برای محاسبه ریشه دوم c باید داشته باشیم $f(x) = x^2-c$



$$t_{i+1} = t_i - \frac{f(t_i)}{f'(t_i)}$$

- شرایط تکنیکی.
  - تابع f(x) باید هموار باشد.
  - تخمین اولیه باید مناسب باشد.

تبدیل یک عدد در مبنای دودویی

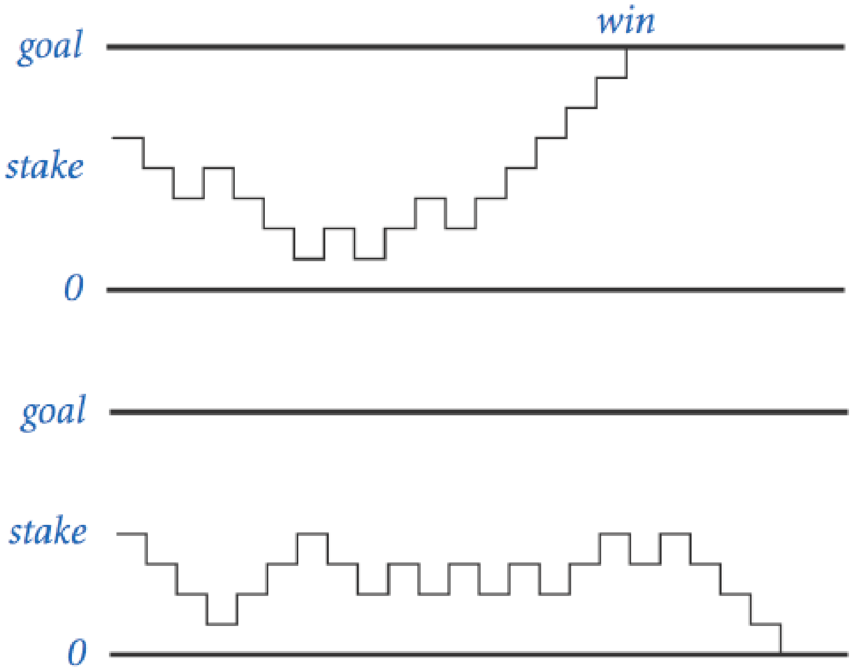
```python
import sys

n = int(sys.argv[1])
v = 1
while v <= n//2:
    v *= 2

while v > 0:
    if n < v:
        print(0, end=‘’)
    else:
        print(1, end=‘’)
        n -= v
    v //= 2

print()
```

```
% python binary.py 19

10011

% python harmonic.py 255

11111111

% python harmonic.py 512

1000000000
```

شبیه‌سازی مونت-کارلو

ورشکستگی قمارباز

- مسئله. یک قمارباز با مبلغ `$stake` شروع می‌کند و هر بار بر روی `$1` شرط می‌بندد تا این که کل پول خود را ببازد یا موجودی وی به مقدار `$goal` برسد.
  - شانس برد چقدر است؟
  - برای بردن به چند شرطبندی نیاز است؟



- یک رویکرد. شبیه‌سازی مونت کارلو
  - شیر یا خط کن و نتیجه را بررسی کن.
  - عمل فوق را تکرار و آمار مورد نیاز را محاسبه کن.

ورشکستگی قمارباز

```python
import sys
import random

stake  = int(sys.argv[1])
goal   = int(sys.argv[2])
trials = int(sys.argv[3])

bets = 0
wins = 0
for t in range(trials):
    cash = stake
    while 0 < cash < goal:
        bets += 1
        if random.randrange(0, 2) == 0:
            cash += 1
        else:
            cash -= 1
    if cash == goal:
        wins += 1

print(str(100 * wins // trials) + ‘% wins’)
print(‘Avg # bets: ’ + str(bets // trials))
```

```
                        stake goal  trials
% python gambler.py 10 20 1000
49% wins
Avg # bets: 99

% python gambler.py 50 250 100
21% wins
Avg # bets: 12712

% python gambler.py 500 2500 100
21% wins
Avg # bets: 1002424
```

شبیه‌سازی و تحلیل نتایج

- حقیقت. احتمال برنده شدن = $stake \div goal$

تفاوت میان موجودی اولیه و هدف

- حقیقت. تعداد مورد انتظار شرط‌بندی‌ها = $stake \times desired\ gain$

- مثال.
  - شانس تبدیل ۵۰۰ دلار به ۲۵۰۰ دلار برابر است با ۲۰ درصد.
  - همچنین تعداد متوسط شرط‌بندی‌ها برابر است با ۱ میلیون!

- ملاحظه. هر دو حقیقت بالا به صورت ریاضی قابل اثبات هستند؛ اما برای سناریوهای پیچیده‌تر، شبیه‌سازی کامپیوتری اغلب بهترین (تنها) روش است.

جریان کنترل: خلاصه

- جریان کنترل.
  - دنباله‌ای از دستورات در برنامه که واقعاً اجرا می‌شوند.
  - دستورات شرطی و حلقه‌ها: تغییر جریان کنترل.

جریان کنترل	توضیحات	مثال‌ها
خط مستقیم	تمام دستورات برنامه به ترتیب داده شده اجرا می‌شوند	

دستورات شرطی	برخی از دستورات برنامه بسته به مقدار بعضی از متغیرهای خاص اجرا می‌شوند	`if`
		`if-else`
		`if-elif`

حلقه‌های تکرار	تا زمانی که شرایط خاصی برقرار باشد، برخی از دستورات به طور مکرر اجرا می‌شوند	`while` `for`