w08-Lec

Iterations Part V

Assembled for 204111
by Areerat Trongratsameethong
Kittipitch Kuptavanich

Basic Loop Programming Techniques

- **Technique 1: Selection within a loop**
- **Technique 2: Input data validation**
- **Technique 3: Interactive loop control**
- **Technique 4: Evaluating equations**

1: Selection within a Loop

```
05 # this program computes the positive and negative sum
06 # of a set of MAXNUMS user entered numbers
07
08 def count_pos_neg():
09
10     MAXNUM = 10
11     pos_total = 0
12     neg_total = 0
13     for i in range(MAXNUM):
14         x = float(input(""))
15         if x > 0:
16             pos_total += x
17         else:
18             neg_total += x
19
20     print("\nThe positive total is ", pos_total)
21     print("\nThe negative total is ", neg_total)
```


2: Input Data Validation

```
08 def verify_month_input():
09     while True:
10         month = int(input("Enter a month between 1 and 12: "))
11         if 1 <= month <= 12:
12             break
13         else:
14             print("Error - the month you entered is not valid.")
15
16     return month
```

3: Interactive Loop Control

```
08 # this program displays a table of numbers,
09 # their squares and cubes
10 # starting from the number 1.
11 # The final number in the table is
12 # input by the user
13
14
15 def display_sqare_and_cube():
16
17     final = int(input("Enter the final number: "))
18
19
20     print("Number Square Cube")
21     print("------------------")
22
23     for num in range(1, final + 1):
24         print("%3d %7d %6d" % (num, num * num, num * num * num))
```

```
Enter the final number: 5
Number Square Cube
------------------
  1       1      1
  2       4      8
  3       9     27
  4      16     64
  5      25    125
```


4: Evaluating Equations

```
02 def equation1(start, end):
03
04     print("x value y value")
05     print("------- -------")
06     x = start
07     while x <= end:
08         y = 10 * (x ** 2) + 3 * x - 2
09         print("%4.2f %10.2f" % (x,  y))
10         x += 0.5
11
12
13 equation1(4, 6)
```

```
x value y value
------- -------
4.00     170.00
4.50     214.00
5.00     263.00
5.50     317.00
6.00     376.00
```


Common Programming Errors

- "Off by one" error, in which the loop executes either one too many or one too few times than intended
  ทำเกินไป 1 Loop หรือ ทำขาดไป 1 Loop (ให้เช็คเงื่อนไขที่ใช้ในการควบคุม Loop ให้ดี)
- Using the assignment operator, =, instead of the equality operator, ==, in the tested expression
  ใช้สัญลักษณ์สำหรับการเปรียบเทียบผิด คือ ใช้ = (สำหรับกำหนดค่าให้กับตัวแปร) แทนที่จะใช้ == (สำหรับเปรียบเทียบ)
- As with the if statement, repetition statements should not use the equality operator, ==, when testing single-precision or double-precision operands
  ไม่ควรใช้การเปรียบเทียบในลักษณะ เท่ากับ == สำหรับการเปรียบเทียบตัวเลขทศนิยม เนื่องจากมีการปัดเศษ
  - (พิจารณาการใช้ฟังก์ชัน ช่วยเปรียบเทียบเช่น `almost_equal()` แทน)

CASE STUDY: FINDING SQUARE ROOT

1: Babylonian Method

Babylonian Method

- **First Documented by Heron of Alexandria**
- **Guess and Check**

สมมติ เราต้องการหารากที่ 2 ของ x $(x > 0)$

1. เริ่มจากการเดาค่า g
2. ถ้า $g * g$ มีค่าใกล้เคียงกับ x มากพอ แสดงว่า g คือคำตอบที่ต้องการ
   - ถ้าไม่ใช่ ให้หาค่า g ใหม่ โดยการหาค่าเฉลี่ยของ g และ x/g
3. ใช้ค่า g ใหม่ที่ได้ แล้วกลับไป test ที่ข้อ 2.

1: Babylonian Method [2]

Consider, for example, finding the square root of 25.

1. **Set g to some arbitrary value, e.g., 3.**
2. **We decide that $3 * 3 = 9$ is not close enough to 25.**
3. **Set g to $\frac{(3 + \frac{25}{3})}{2} = 5.67$**
4. **We decide that $5.67 * 5.67 = 32.15$ is still not close enough to 25.**
5. **Set g to $\frac{(5.67 + \frac{25}{5.67})}{2} = 5.04$**
6. **We decide that $5.04 * 5.04 = 25.4$ is close enough, so we stop and declare 5.04 to be an adequate approximation to the square root of 25.**

1: Babylonian Method [3]

```
03 def sqrt1(x):
04     g = x / 2
05     epsilon = 0.00001
06
07     while abs(g * g - x) > epsilon:
08         print("g = ", g)
09         g = (g + (x / g)) / 2
10
11     return g
12
13 x = 3
14 print("sqrt {0:f} = {1:f}".format(x, sqrt1(x)))
```

```
g =  1.5
g =  1.75
g =  1.7321428571428572
sqrt 3.000000 = 1.732051
```


2: Exhaustive Enumeration

```
01 epsilon = 0.01
02
03 def sqrt2(x):
04
05     step = epsilon ** 2
06     num_guesses = 0
07     ans = 0.0
08     while abs(ans ** 2 - x) >= epsilon and ans <= x:
09         ans += step
10         num_guesses += 1
11     print('num_guesses =', num_guesses)
12
13     if abs(ans ** 2 - x) >= epsilon:
14         print('Failed on square root of', x)
15         return None
16     else:
17         print(ans, 'is close to square root of', x)
18         return ans
19
20 x = 25
21 ans = sqrt2(x)
```

```
num_guesses = 49990
4.999000000001688 is close to square root of 25
```

- ไล่ลองค่าที่เป็นไปได้ทั้งหมดจนกว่าจะเจอคำตอบ
  - เป็น **guess and check** แบบหนึ่ง
  - เดาแล้วเพิ่มค่าทีละ `step` จนเจอคำตอบ

2: Exhaustive Enumeration [2]

```
num_guesses = 49990
4.999000000001688 is close to square root of 25
```

- สังเกตว่าค่าที่คำนวณได้ ไม่ได้เท่ากับ 5
  - The program did what it was intended to do.
  - Printing out 5 is no better than any value close enough to 5
- ถ้า x = 0.25?
  - ไม่เจอ solution เนื่องจากหาคำตอบในช่วง 0.... x
  - Solution คือ 0.5
- Exhaustive Enumeration ใช้ได้ต่อเมื่อคำตอบอยู่ในช่วงค่าที่ต้องการหา

2: Exhaustive Enumeration [3]

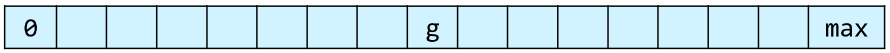
- Let's try something bigger. e.g. x = 123456

```
num_guesses = 3513631
Failed on square root of 123456
```

  - Why???
  - `step` ใหญ่เกินไป ทำให้ข้ามคำตอบที่เป็นไปได้
  - ถ้าใช้ `step` เป็น `epsilon ** 3 = 0.000001`
  - พบคำตอบแต่ใช้เวลานาน (`351,000,000 guesses`)

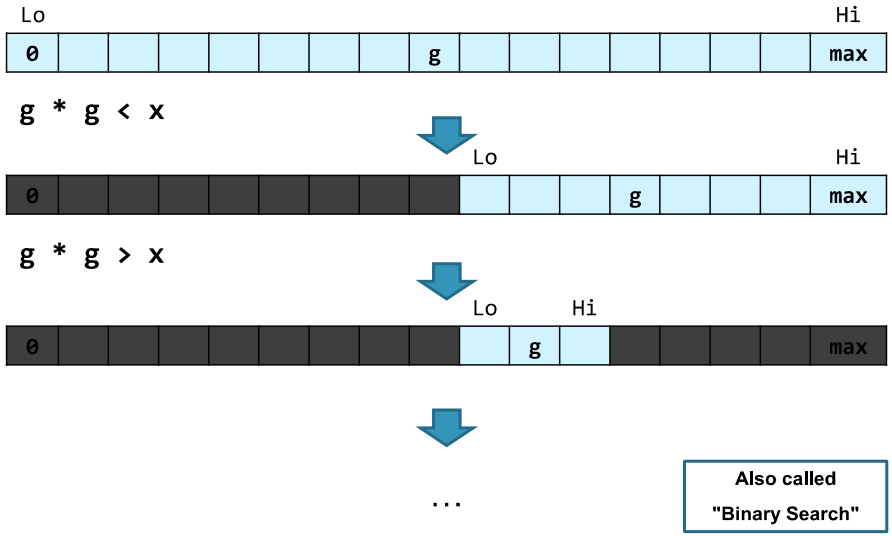
3: Bisection Search

- จาก Exhaustive Enumeration มีข้อสังเกตคือ ค่าแต่ละค่าในช่วงที่ต้องการหาเป็นค่าที่เรียงกัน
  - ปัญหานี้มีลักษณะคล้ายการเปิดหาคำศัพท์ใน Dictionary – โดยปกติเราทำอย่างไร?
  - ในทางปฏิบัติ - หากเปิดมาหน้าที่อยู่ในช่วงตัวอักษร*ก่อน*คำที่ต้องการหา
    - แสดงว่าหน้านั้นและ ทุกหน้าก่อนนั้นไม่มีคำตอบ หาคำตอบในหน้าหลังจากนั้นเท่านั้น



  - เดา g ที่ครึ่งหนึ่ง ถ้า `g * g < x` แสดงว่า g มีค่าน้อยไป
  - คำตอบอยู่ในช่วงทางด้านขวาของ g (ตัดช่วงที่ไม่ใช่คำตอบทิ้งทีละครึ่ง)

3: Bisection Search



3: Bisection Search [3]

```
04 def sqrt3(x):
05     num_guesses = 0
06     low = 0.0
07     high = max(1.0, x)
08     ans = (high + low) / 2.0
09     while abs(ans ** 2 - x) >= epsilon:
10         print('low =', low, 'high =', high, 'ans =', ans)
11         num_guesses += 1
12         if ans ** 2 < x:
13             low = ans
14         else:
15             high = ans
16
17         ans = (high + low) / 2.0
18     print('num_guesses =', num_guesses)
19     print(ans, 'is close to square root of', x)
20
21 x = 25
22 ans = sqrt3(x)
```

```
low = 0.0 high = 25 ans = 12.5
…
low = 4.99267578125 high = 5.0048828125 ans = 4.998779296875
low = 4.998779296875 high = 5.0048828125 ans = 5.0018310546875
num_guesses = 13
5.00030517578125 is close to square root of 25
```

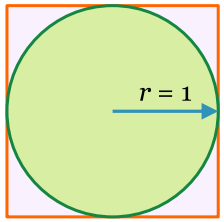

CASE STUDY: MONTE CARLO METHOD

Estimating Pi

พื้นที่ของวงกลมในรูปเท่ากับ_____

พื้นที่ของสี่เหลี่ยมจัตุรัสในรูปเท่ากับ_____

หากปาลูกดอกไปที่รูปในลักษณะ Random

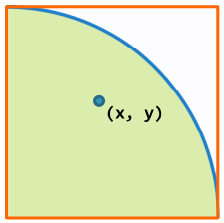


$$\frac{\text{โอกาสที่จะลูกดอกจะตกในกรอบวงกลม}}{\text{โอกาสที่ลูกดอกจะตกภายในกรอบสี่เหลี่ยม}} = \text{\_\_\_\_\_} \approx \frac{\text{จำนวนลูกดอกที่ตกภายในวงกลม}}{\text{จำนวนลูกดอกที่ตกภายในสี่เหลี่ยม}}$$

ดังนั้นจะได้ว่า $\pi \approx \frac{\text{จำนวนลูกดอกที่ตกภายในวงกลม}}{\text{จำนวนลูกดอกที่ตกภายในสี่เหลี่ยม}} \times$ _____

Estimating Pi [2]

- **ทำปัญหาให้ง่ายขึ้นด้วยการพิจารณาแค่ 1 quadrant**
- **เราจะทราบได้อย่างไรว่า** (x, y) **อยู่ในหรือนอกส่วนของวงกลม (sector)?**
  - **Pythagoras ถ้า** (x, y) **อยู่ในวงกลม** $\sqrt{x^2 + y^2}$ **ต้อง** ≤ 1
    - **หรือ** $x^2 + y^2 \leq 1$
- **สุ่ม** $x\ (0 \leq x \leq 1)$ **และ** $y\ (0 \leq y \leq 1)$ **ด้วย** `random.random()`



Estimating Pi [3]

```
01 import random
02
03 def rand_num_in_range(lo, hi):
04     return lo + (hi - lo) * random.random()
05
06 def is_in_circle(x, y):
07     return x ** 2 + y ** 2 <= 1
08
09 def find_pi(sample):
10     num_in_circle = 0
11     for s in range(sample):
12         x = rand_num_in_range(0, 1)
13         y = rand_num_in_range(0, 1)
14         if is_in_circle(x, y):
15             num_in_circle += 1
16     return (num_in_circle / sample) * 4
17
18 print(find_pi(10000))
```


Monte Carlo Methods

ใช้ **random number** เพื่อแก้ปัญหา

General approach

- ทำการจำลองเหตุการณ์หลาย ๆ เหตุการณ์
  - ในแต่ละครั้ง ทำการสุ่มค่า เช่น โยนเหรียญ ทอดลูกเต๋า สุ่มพิกัด
  - นับจำนวนที่สำเร็จ
- คำนวณค่าที่ได้
  - **Expected Odds ≈ Observed Odds = (successful trials) / (total trials)**

Law of Large Numbers:

- จำนวนครั้งที่มากขึ้น = ค่าที่ใกล้เคียงค่าที่ถูกต้องขึ้น = ใช้เวลามากขึ้น

Practice 1

ให้นำเข้าข้อมูลประเภทจำนวนเต็ม 2 ค่า คือค่าแรก (first) และค่าสุดท้าย (last) ผ่านทาง keyboard และนับว่ามีจำนวนเฉพาะกี่จำนวน อะไรบ้างเช่น

7 ถึง 21: มี จำนวนเฉพาะ 5 ตัวคือ: 7, 11, 13, 17, 19

- ตัวอย่างการ run 1

```
Input first value: 7
Input last value: 21
The prime number(s) between 7 - 21 are
7 11 13 17 19
Total: 5 numbers
```


Practice 2

ตัวอย่าง Output

```
Average of Sale Amount of Week (1) = 12800.00
Average of Sale Amount of Week (2) = 13050.00
Average of Sale Amount of Week (3) = 13728.57
Average of Sale Amount of Week (4) = 14500.00
Average of Sale Amount for All = 13519.64
```

ให้เขียนโปรแกรมภาษา Python เพื่อหาค่าเฉลี่ยของยอดขายทั้งเดือน ค่าเฉลี่ยของยอดขายต่อสัปดาห์ ให้สอดคล้องกับตัวอย่างการ run โปรแกรมด้านล่าง โดยที่ค่าข้อมูลให้นำเข้าผ่านทาง keyboard

Week	Su	Mo	Tu	We	Th	Fr	Sa	Average (Per Week)
1	15000.00	10000.00	11500.00	9500.00	12600.00	14500.00	16500.00	12800.00
2	14500.00	11000.00	12500.00	9600.00	12650.00	14550.00	16550.00	13050.00
3	14800.00	11500.00	13500.00	9800.00	13500.00	15500.00	17500.00	13728.57
4	16000.00	12450.00	13550.00	10500.00	14500.00	16000.00	18500.00	14500.00
						Average (All)		13519.64

Practice 3

ให้นำเข้าข้อมูลจำนวนแถว (row) และ พิมพ์ผลลัพธ์ดังตัวอย่างแสดงด้านล่าง

- ตัวอย่างการ **run 1**

```
Input row: 3
1
1 2
1 2 3
```

- ตัวอย่างการ **run 2**

```
Input row: 4
1
1 2
1 2 3
1 2 3 4
```


Practice 4

ให้นำเข้าข้อมูลจำนวนแถว (row) และ พิมพ์ผลลัพธ์ดังตัวอย่างแสดงด้านล่าง

- ตัวอย่างการ **run 1**

```
Input row: 3
* * 1
* 2 2
3 3 3
```

- ตัวอย่างการ **run 2**

```
Input row: 4
* * * 1
* * 2 2
* 3 3 3
4 4 4 4
```


Practice 5

ให้นำเข้าข้อมูลจำนวนแถว (row) และ พิมพ์ผลลัพธ์ดังตัวอย่างแสดงด้านล่าง

- ตัวอย่างการ **run 1**

```
Input row: 3
* * *
* *
*
```

- ตัวอย่างการ **run 2**

```
Input row: 4
* * * *
* * *
* *
*
```


References

- http://www.mathpages.com/home/kmath190.htm
- http://www.cs.cmu.edu/~112/notes/notes-monte-carlo.html
- Gary J. Bronson *A First Book of ANSI C, Fourth Edition*
- Guttag, John V. *Introduction to Computation and Programming Using Python, Revised*