

Big Oh

Big Oh คืออะไร

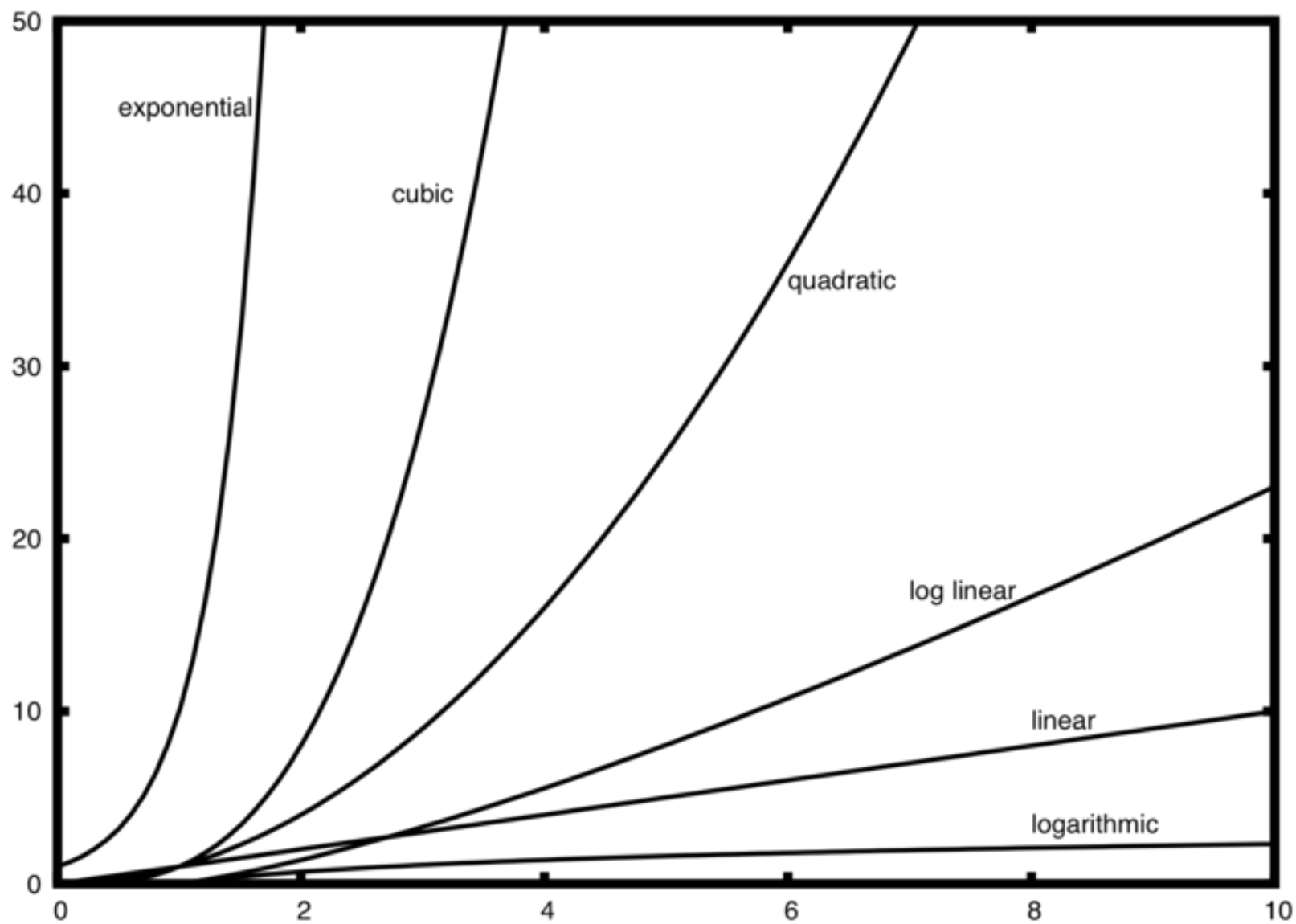
- ตัววัดประสิทธิภาพการทำงาน
- วัดตามจำนวนข้อมูล n
- ตัวอย่าง **sum of n**
 - จำนวน **step** ในการทำงาน $T(n) = 2 + n$
 - $O(n)$
 - ดูตัวที่ยกกำลังเยอะสุด
- $T(n) = 5n^2 + 3n$
 - $O(n^2)$
 - ดูตัวที่ยกกำลังเยอะสุดและไม่ต้องดูสัมประสิทธิ์

```
1 n = 10
2 thesum = 0
3 for i in range(1, n + 1) :
4     thesum = thesum + i
5 print(thesum)
```

ชื่อเรียก

f(n)	Name
1	Constant
$\log n$	Logarithmic
n	Linear
$n \log n$	Log Linear
n^2	Quadratic
n^3	Cubic
2^n	Exponential

กราฟอัตราการโต



Big Oh สำคัญอย่างไร?

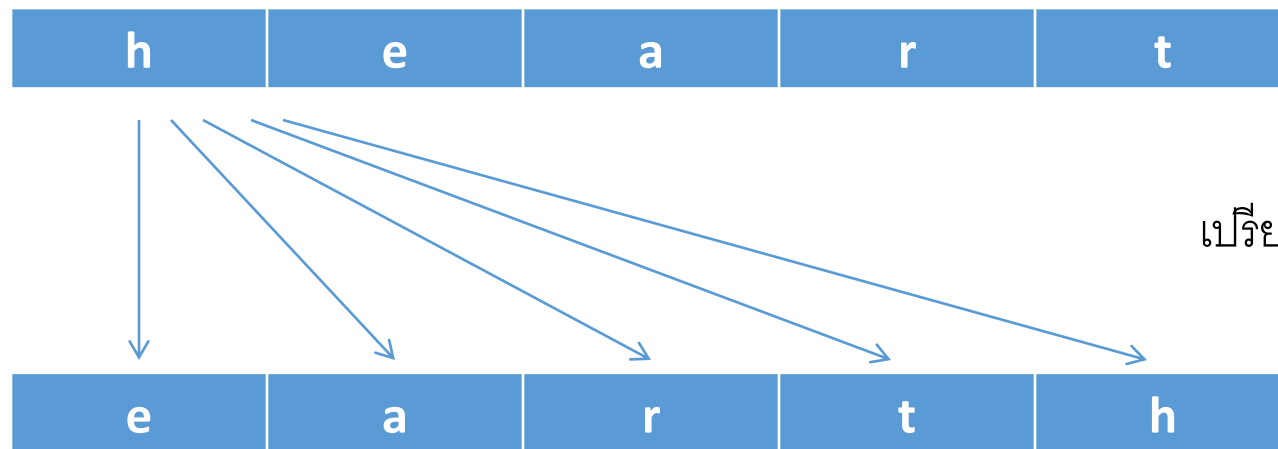
- รู้ว่าประสิทธิภาพของแต่ละ **data structure** เป็นอย่างไร
- เลือก **data structure** ที่มีประสิทธิภาพเหมาะกับปัญหา
- คิด **algorithm** ที่จะทำให้ใช้ **data structure** ได้อย่างมีประสิทธิภาพ

Anagram detection

- String 2 ตัวประกอบด้วยชุดตัวอักษรเดียวกันหรือไม่
- heart กับ earth → true
- python กับ typhon → true
- แก้ปัญหาได้หลายวิธี O ต่างกัน

Check off

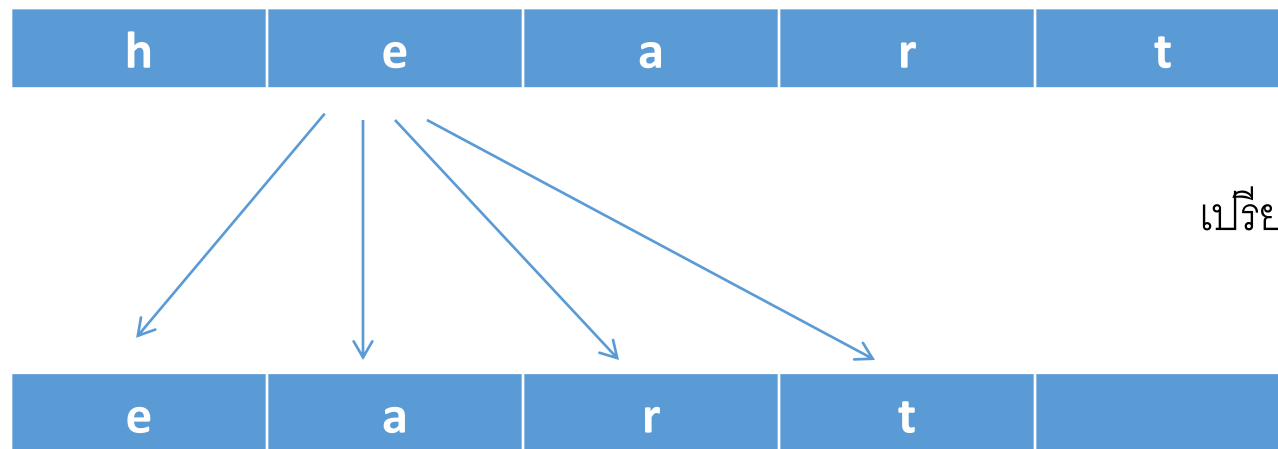
- ทดสอบทุกคู่



เปรียบเทียบ **n** ครั้ง

Check off

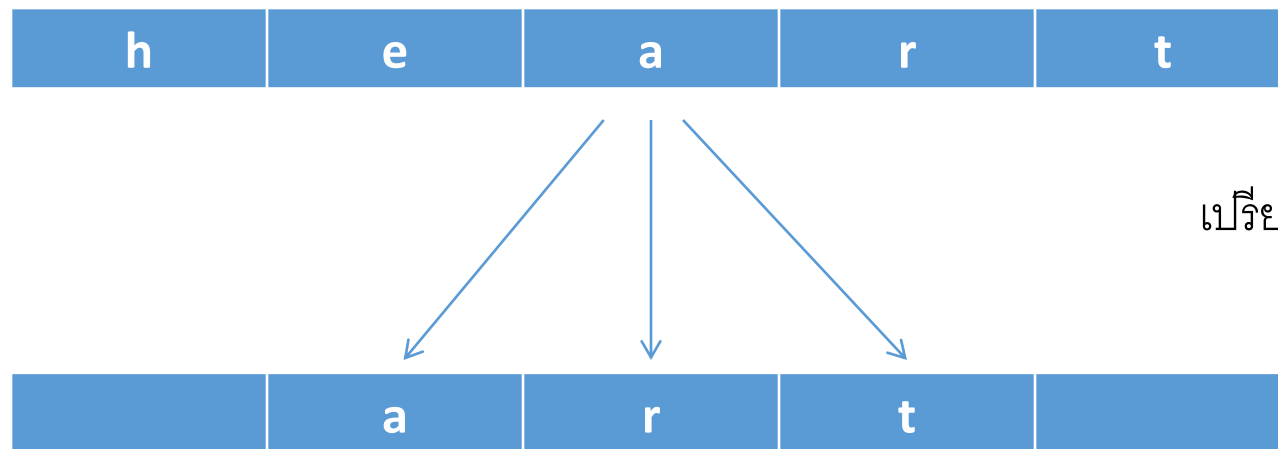
- ทดสอบทุกคู่



เปรียบเทียบ **n-1** ครั้ง

Check off

- ทดสอบทุกคู่



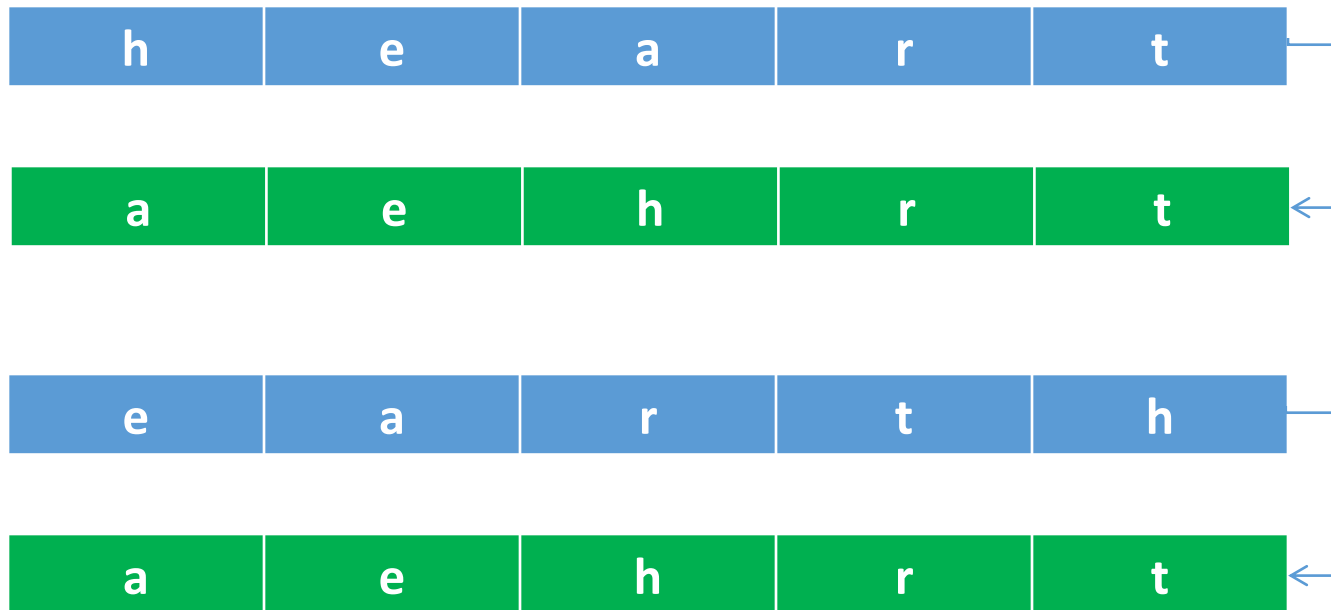
เปรียบเทียบ $n-2$ ครั้ง

Check off

- $n + (n - 1) + (n - 2) + \dots + 1$
- ออนุกรมเลขคณิต (ม. ปลาย)
- $n * (n + 1) / 2$
- $T(n) = \frac{1}{2} * n^2 + \frac{1}{2} * n = O(n^2)$

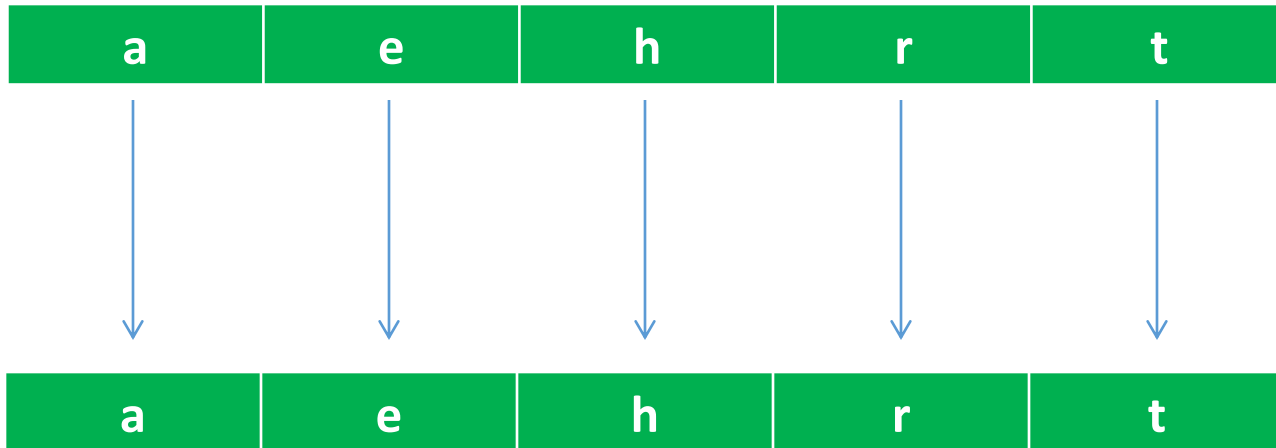
Sort & compare

- เรียงลำดับแล้วเปรียบเทียบเป็นคู่ๆ



Sort & compare

- เรียงลำดับแล้วเปรียบเทียบเป็นคู่ๆ



Sort & compare

- เรียงลำดับ
 - $O(n \log n)$
 - $O(n^2)$
- เปรียบเทียบเป็นคู่ๆ
 - $O(n)$
- รวม $O(n \log n) + O(n) \rightarrow O(n \log n)$

Count & compare

- นับแล้วเปรียบเทียบ
- ตัวอักษร^๓มีทั้งหมด 26 ตัว

a	b	c	d	e	f	g	h	i	j
k	l	m	n	o	p	q	r	s	t
u	v	w	x	y	z				

Count & compare

- นับว่าแต่ละ **string** มีตัวอักษรตัวไหนเท่าไร
- นับ heart

a(1)	b	c	d	e(1)	f	g	h(1)	i	j
k	l	m	n	o	p	q	r(1)	s	t(1)
u	v	w	x	y	z				

- นับ earth

a(1)	b	c	d	e(1)	f	g	h(1)	i	j
k	l	m	n	o	p	q	r(1)	s	t(1)
u	v	w	x	y	z				

Count & compare

- เปรียบเทียบตัวอักษรเป็นคู่ๆ

heart

a(1)	b	c	d	e(1)	f	g	h(1)	i	j
k	l	m	n	o	p	q	r(1)	s	t(1)
u	v	w	x	y	z				

earth

a(1)	b	c	d	e(1)	f	g	h(1)	i	j
k	l	m	n	o	p	q	r(1)	s	t(1)
u	v	w	x	y	z				

Count & compare

- เปรียบเทียบตัวอักษรเป็นคู่ๆ

heart

a(1)	b	c	d	e(1)	f	g	h(1)	i	j
k	l	m	n	o	p	q	r(1)	s	t(1)
u	v	w	x	y	z				

earth

a(1)	b	c	d	e(1)	f	g	h(1)	i	j
k	l	m	n	o	p	q	r(1)	s	t(1)
u	v	w	x	y	z				

Count & compare

- เปรียบเทียบตัวอักษรเป็นคู่ๆ

heart

a(1)	b	c	d	e(1)	f	g	h(1)	i	j
k	l	m	n	o	p	q	r(1)	s	t(1)
u	v	w	x	y	z				

earth

a(1)	b	c	d	e(1)	f	g	h(1)	i	j
k	l	m	n	o	p	q	r(1)	s	t(1)
u	v	w	x	y	z				

Count & compare

- นับตัวอักษร
 - $T(n) = n * 2$
- เปรียบเทียบตัวอักษรเป็นคู่ๆ
 - $T(n) = 26$
- รวม $T(n) = n * 2 + 26 \rightarrow O(n)$

แบบฝึกหัด

- <http://interactivepython.org/runestone/static/pythonds/AlgorithmAnalysis/AnAnagramDetectionExample.html>

List

Operation	Big-O Efficiency
index []	O(1)
index assignment	O(1)
append	O(1)
pop()	O(1)
pop(i)	O(n)
insert(i,item)	O(n)
del operator	O(n)
iteration	O(n)
contains (in)	O(n)
get slice [x:y]	O(k)
del slice	O(n)
set slice	O(n+k)
reverse	O(n)
concatenate	O(k)
sort	O(n log n)
multiply	O(nk)

slice

```
1 L = ['a','b','c','d','e','f']
2 print(L[3:5])
3 L[3:5] = []
4 print(L)
```

```
['d', 'e']
['a', 'b', 'c', 'f']
```

Dictionary

operation	Big-O Efficiency
copy	$O(n)$
get item	$O(1)$
set item	$O(1)$
delete item	$O(1)$
contains (in)	$O(1)$
iteration	$O(n)$

Linear structure

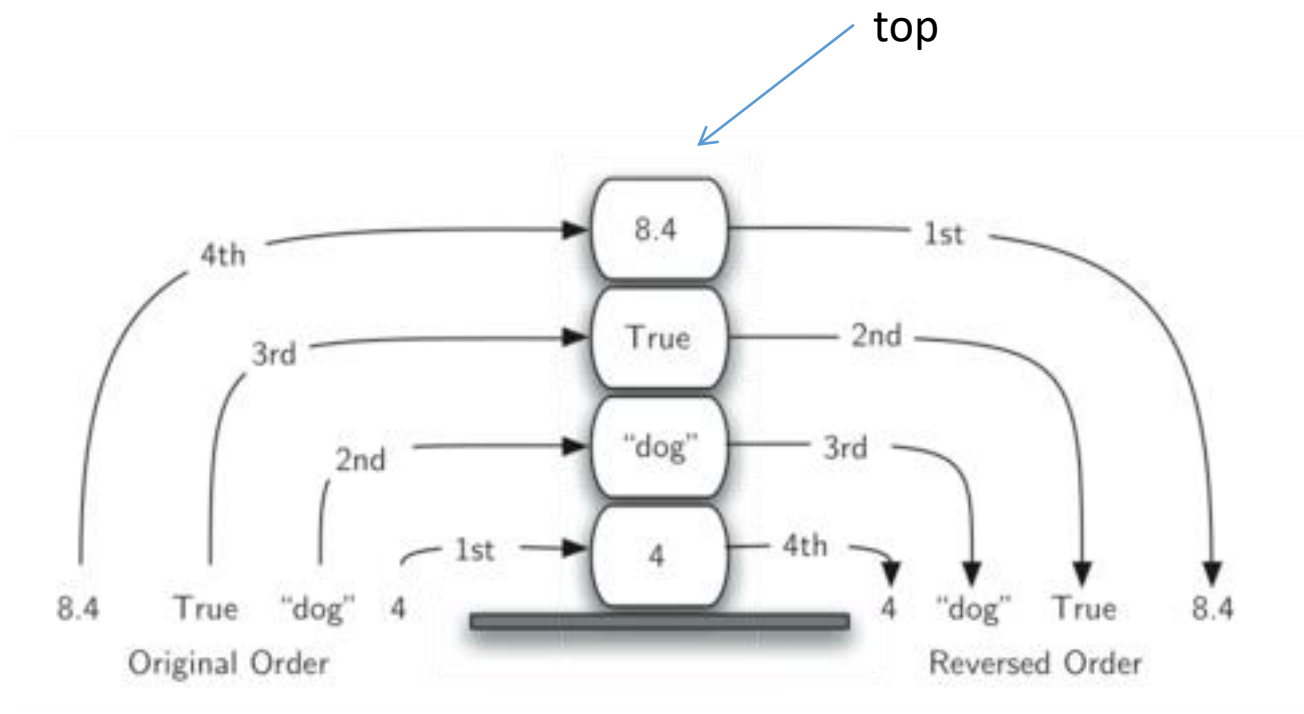
Linear structure คือ?

- โครงสร้างมีลำดับ
- มีหัว มีท้าย หรือ มีบน มีล่าง
- list, stack, queue
 - ต่างกันตรงวิธีเพิ่มและลบข้อมูล



Stack

- มีลำดับแบบ **last-in-first-out**
- ลำดับที่สำคัญคือ **top**



Stack ADT

- `stack` มีการทำงานอย่างไร ควรจะมี `method` อะไรบ้าง
- `Stack()` สร้าง `stack` ว่างๆ และ `return stack` มาให้
- `push(item)` เพิ่ม `item` และ `ไม่ return`
- `pop()` เอา `item` ที่อยู่บนสุดออกมาและ `return item` นั้น
- `peek()` `return item` ที่อยู่บนสุด
- `isEmpty()` `return` ค่าว่า `stack` ว่างหรือไม่
- `size()` `return` จำนวน `item` ใน `stack`

Stack ADT

Stack Operation	Stack Contents	Return Value
s.isEmpty()	[]	True
s.push(4)	[4]	
s.push('dog')	[4,'dog']	
s.peek()	[4,'dog']	'dog'
s.push(True)	[4,'dog',True]	
s.size()	[4,'dog',True]	3
s.isEmpty()	[4,'dog',True]	False
s.push(8.4)	[4,'dog',True,8.4]	
s.pop()	[4,'dog',True]	8.4
s.pop()	[4,'dog']	True
s.size()	[4,'dog']	2

Stack's implementation

```
1 class Stack:
2     def __init__(self):
3         self.items = []
4
5     def isEmpty(self):
6         return self.items == []
7
8     def push(self, item):
9         self.items.append(item)
10
11    def pop(self):
12        return self.items.pop()
13
14    def peek(self):
15        return self.items[len(self.items)-1]
16
17    def size(self):
18        return len(self.items)
19
```

การใช้งาน stack

Run

```
1 from pythonds.basic.stack import Stack
2
3 s=Stack()
4
5 print(s.isEmpty())
6 s.push(4)
7 s.push('dog')
8 print(s.peek())
9 s.push(True)
10 print(s.size())
11 print(s.isEmpty())
12 s.push(8.4)
13 print(s.pop())
14 print(s.pop())
15 print(s.size())
16
```

True
dog
3
False
8.4
True
2

Stack in action

Python 2.7

```
7
8     def push(self, item):
9         self.items.insert(0,item)
10
11     def pop(self):
12         return self.items.pop(0)
13
14     def peek(self):
15         return self.items[0]
16
17     def size(self):
18         return len(self.items)
19
20 → s = Stack()
21 s.push('hello')
22 s.push('true')
23 print(s.pop())
```

<< First

< Back

Step 2 of 17

Forward >

Last >>

→ line that has just executed

→ next line to execute

Frames

Global frame
Stack

Objects

Stack class
hide attributes

__init__	function __init__(self)
isEmpty	function isEmpty(self)
peek	function peek(self)
pop	function pop(self)
push	function push(self, item)
size	function size(self)

Balanced parentheses

- วงเล็บเปิดวงเล็บปิดถูกหรือไม่ (สมมติว่ามีแค่วงเล็บ)
- ตัวอย่างที่ถูก
 - (() () () ())
 - (((())))
 - (() ((()) ()))
- ตัวอย่างที่ผิด
 - ((((((
 -)))
 - ((()((

Implementation

```
1 from pythonds.basic.stack import Stack
2
3 def parChecker(symbolString):
4     s = Stack()
5     balanced = True
6     index = 0
7     while index < len(symbolString) and balanced:
8         symbol = symbolString[index]
9         if symbol == "(":
10             s.push(symbol) ← เจอ ( ยัดเข้า stack
11         else:
12             if s.isEmpty():
13                 balanced = False ← stack ว่างเจอ ) ไม่ balanced แน่นนอน
14             else:
15                 s.pop() ← เจอ ) pop ( ออก
16
17         index = index + 1
18
19     if balanced and s.isEmpty():
20         return True
21     else:
22         return False
```


Balanced symbols

- $[\{ () \})$ ถูกหรือไม่
- ตัวอย่างที่ถูก
 - $\{ \{ ([] []) \} () \}$
 - $[[\{ \{ (()) \} \}]]$
 - $[] [] [] () \{ \}$
- ตัวอย่างที่ผิด
 - $([]]$
 - $((()]))$
 - $[\{ ()]$

Implementation

```
1 def parChecker(symbolString):
2     s = Stack()
3     balanced = True
4     index = 0
5     while index < len(symbolString) and balanced:
6         symbol = symbolString[index]
7         if symbol in "([{":
8             s.push(symbol)
9         else:
10            if s.isEmpty():
11                balanced = False
12            else:
13                top = s.pop()
14                if not matches(top, symbol):
15                    balanced = False
16            index = index + 1
17    if balanced and s.isEmpty():
18        return True
19    else:
20        return False

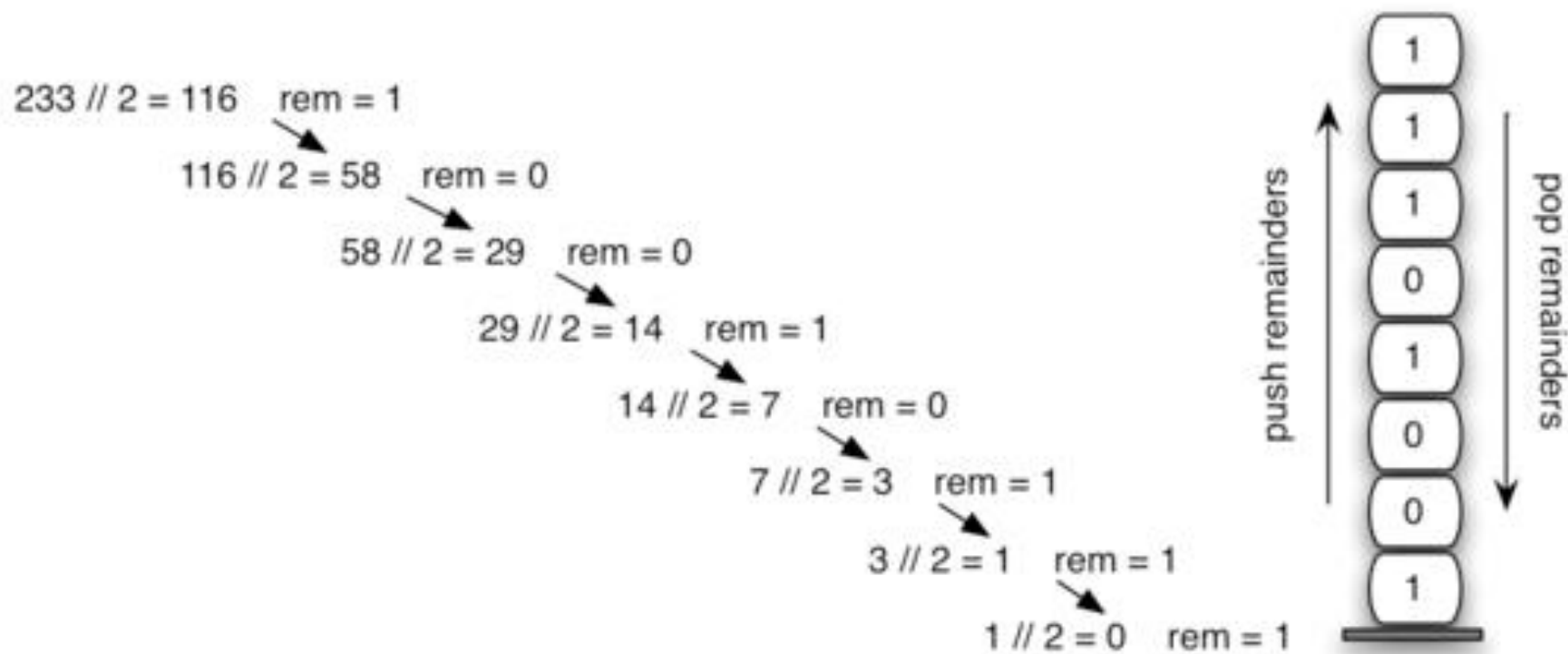
1 def matches(open, close):
2     opens = "([{"
3     closers = ")]}"
4     return opens.index(open) == closers.index(close)
```

แปลงเลขฐาน 2

- แปลงจากฐาน 10 ไปเป็นฐาน 2
- ตัวอย่างฐาน 10 แยกหลัก
 - 512 หาร 10 เศษ 2 ยัดลง stack
 - ผลหาร 51 หาร 10 เศษ 1 ยัดลง stack
 - ผลหาร 5 หาร 10 เศษ 5 ยัดลง stack
 - pop ออกมาจนหมด

5
1
2

แปลงเลขฐาน 2



Implementation

```
1 from pythonds.basic.stack import Stack
2
3 def divideBy2(decNumber):
4     remstack = Stack()
5
6     while decNumber > 0:
7         rem = decNumber % 2
8         remstack.push(rem)
9         decNumber = decNumber // 2
10
11     binString = ""
12     while not remstack.isEmpty():
13         binString = binString + str(remstack.pop())
14
15     return binString
16
17 print(divideBy2(42))
```

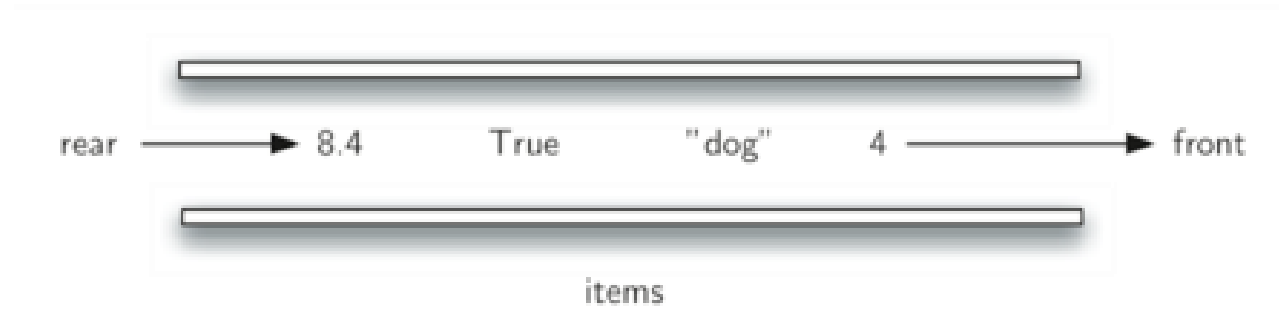
← หาร 2 ยัดเศษลง stack

แปลงเลขฐานอื่นๆ

- ลองทำดู
- เฉลย
 - <http://interactivepython.org/runestone/static/pythonds/BasicDS/ConvertingDecimalNumberstoBinaryNumbers.html>

Queue

- มีลำดับแบบ **first-in-first-out**
- ลำดับที่สำคัญคือ **rear** กับ **front**



Queue ADT

- Queue() สร้าง queue ว่างๆ และ return queue มาให้
- enqueue(item) เพิ่ม item และไม่ return
- dequeue() เอา item ที่อยู่บนหน้าออกมาและ return item นั้น
- isEmpty() return ค่าว่า queue ว่างหรือไม่
- size() return จำนวน item ใน queue

Queue ADT

Queue Operation	Queue Contents	Return Value
q.isEmpty()	[]	True
q.enqueue(4)	[4]	
q.enqueue('dog')	['dog',4]	
q.enqueue(True)	[True,'dog',4]	
q.size()	[True,'dog',4]	3
q.isEmpty()	[True,'dog',4]	False
q.enqueue(8.4)	[8.4,True,'dog',4]	
q.dequeue()	[8.4,True,'dog']	4
q.dequeue()	[8.4,True]	'dog'
q.size()	[8.4,True]	2

Queue's implementation

```
class Queue:
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return self.items == []

    def enqueue(self, item):
        self.items.insert(0,item)

    def dequeue(self):
        return self.items.pop()

    def size(self):
        return len(self.items)
```

Queue in action

```
Python 2.7
6     return self.items == []
7
8     def enqueue(self, item):
9         self.items.insert(0,item)
10
11     def dequeue(self):
12         return self.items.pop()
13
14     def size(self):
15         return len(self.items)
16
17 q=Queue()
18
19 q.enqueue(4)
20 q.enqueue('dog')
21 q.enqueue(True)
22 print(q.size())
```

<< First

< Back

Step 13 of 21

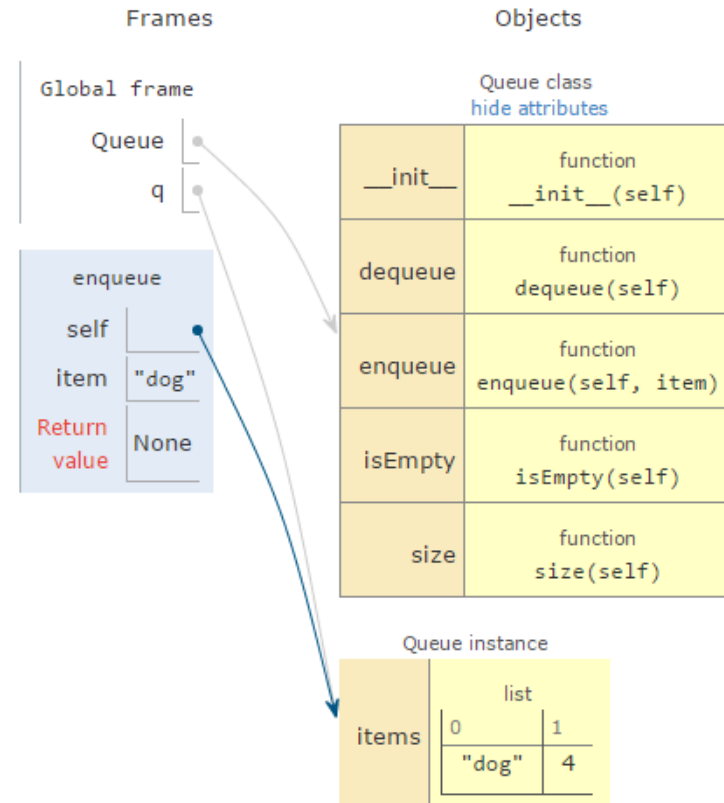
Forward >

Last >>

→ line that has just executed

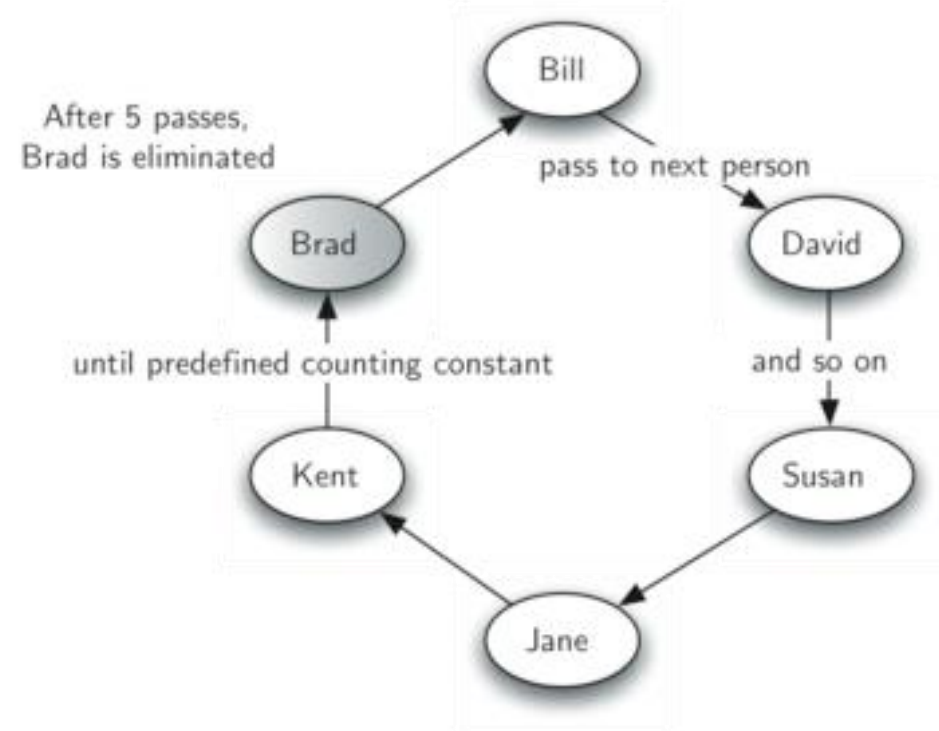
→ next line to execute

Visualized using Online Python Tutor by Philip Guo



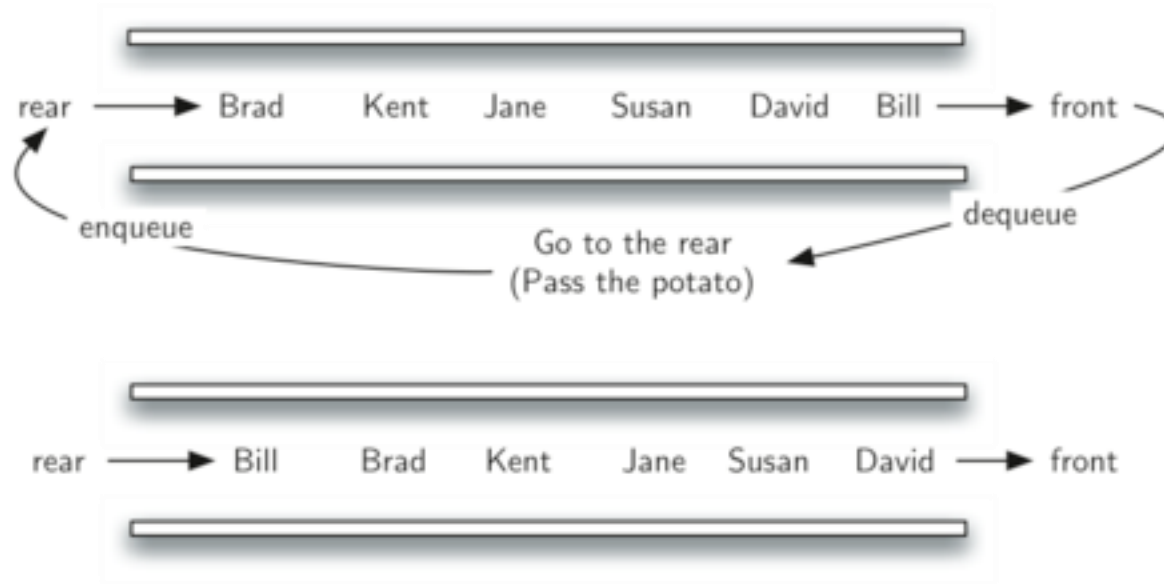
Hot potato

- ส่ง **potato** ไปให้คนข้างๆ หยุดที่ใครคนนั้นออกจากเกม คนสุดท้ายชนะ
- เล่นแบบง่ายๆ ส่งเป็นจำนวนครั้งแล้วหยุด
 - ในตัวอย่างนี้ส่ง 5 ครั้ง



คิดแบบ queue

- คนที่อยู่หัว queue คือคนที่ถือ potato
- เมื่อส่ง potato ให้คนข้างๆ ก็ไปต่อท้าย queue



Implementation

```
1 from pythonds.basic.queue import Queue
2
3 def hotPotato(namelist, num):
4     simqueue = Queue()
5     for name in namelist:
6         simqueue.enqueue(name)
7
8     while simqueue.size() > 1:
9         for i in range(num):
10             simqueue.enqueue(simqueue.dequeue())
11
12         simqueue.dequeue()
13
14     return simqueue.dequeue()
15
16 print(hotPotato(["Bill", "David", "Susan", "Jane", "Kent", "Brad"], 7))
17
```

เริ่มใส่ชื่อคนเล่น

ส่งเป็นจำนวนครั้ง คนที่ส่งแล้วไปต่อท้าย

ครบจำนวนใครอยู่หน้า queue (ถือ potato) ถูกกำจัด

เหลือคนสุดท้ายคือผู้ชนะ

Printing task

- จำลองสถานการณ์เพื่อคำนวณเวลารอในการพิมพ์
- ความสามารถของ **printer** คือ จำนวนหน้าต่อนาที (ppm)
- ความน่าจะเป็นของการเกิด **task = 1** งานใน **180** วินาที
 - สุ่ม **1** ใน **180** ทุกๆ **1** วินาที (tick)
- จำนวนหน้าในแต่ละงาน **1 – 20** หน้าโอกาสเท่ากันหมด

Printer class

- Variables
 - ppm
 - Task ปัจจุบัน
 - เวลาที่เหลือในการทำ **task** ปัจจุบัน
- Methods
 - ตรวจสอบการทำงานทุกๆ 1 tick
 - **Printer** ทำงานอยู่หรือไม่ (**busy**)
 - เริ่ม **task** ใหม่

Task class

- Variables

- เวลาที่ส่งงานเข้า queue
- จำนวนหน้า

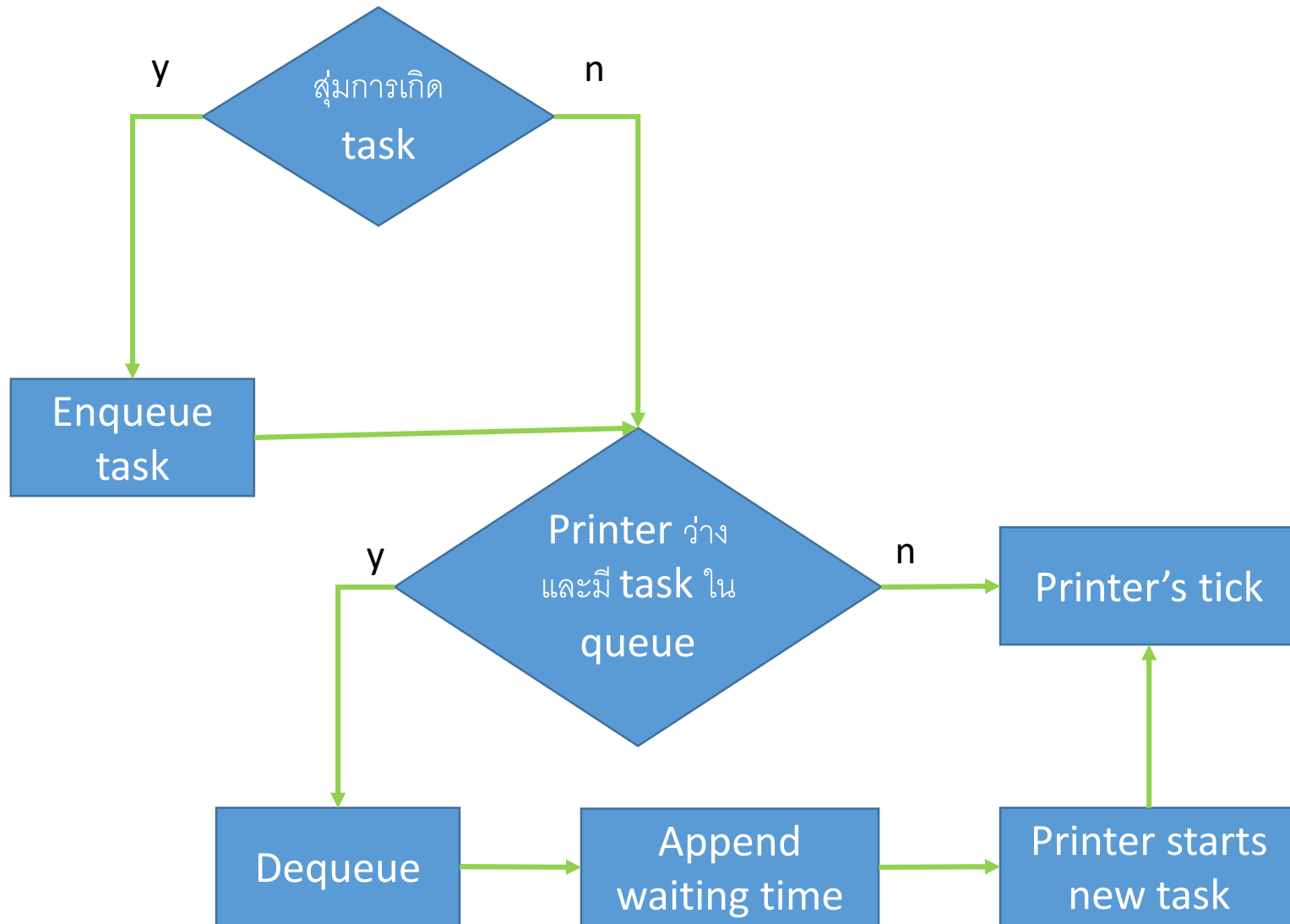
- Method

- คำนวณเวลาที่รอ queue

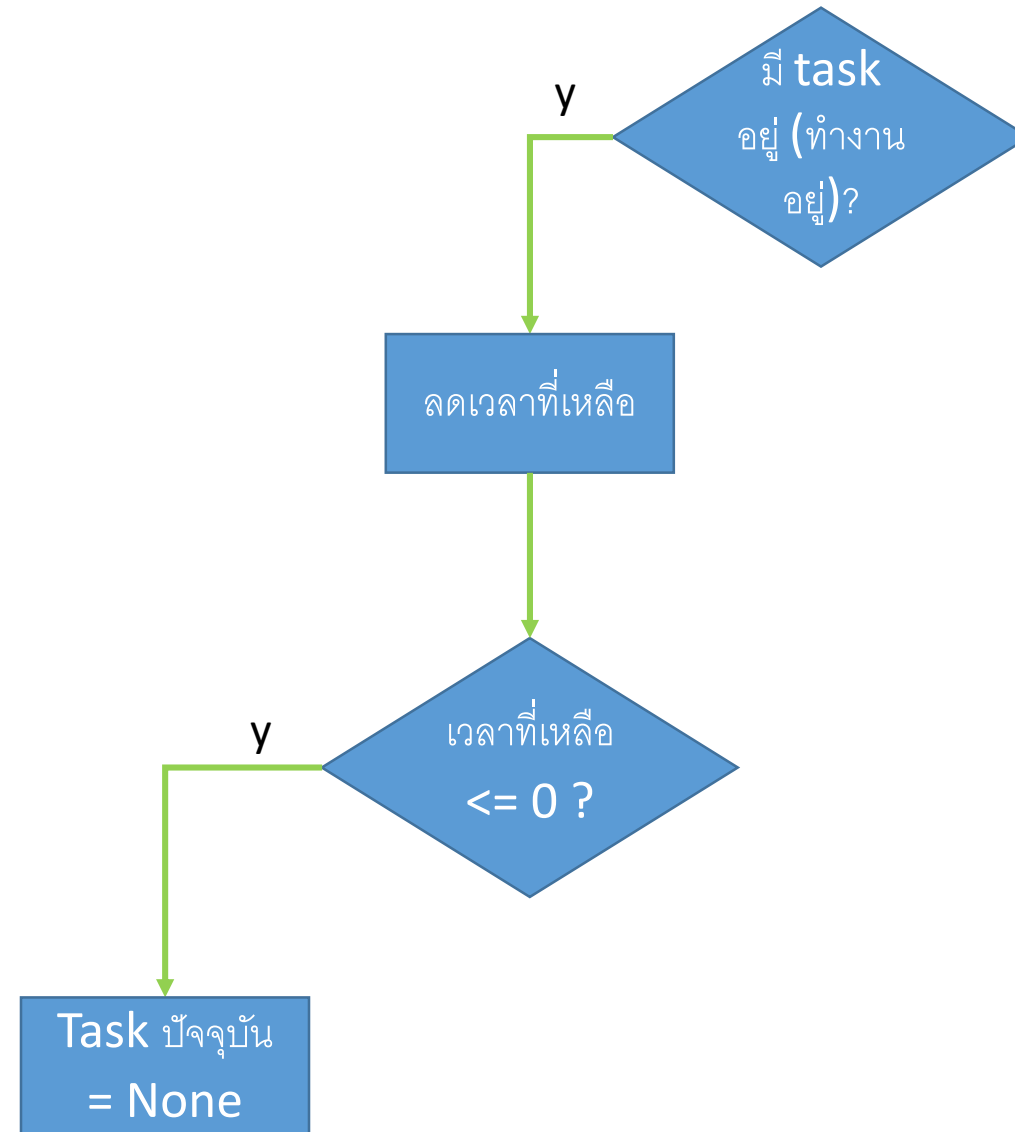
Random

- สุ่มตัวเลข
- `random.randrange(n, m)`
 - สุ่มจำนวนเต็มระหว่าง n ถึง $m - 1$

Simulation



Printer's tick



Printer class implementation

```
1  class Printer:
2      def __init__(self, ppm):
3          self.pagerate = ppm
4          self.currentTask = None
5          self.timeRemaining = 0
6
7      def tick(self):
8          if self.currentTask != None:
9              self.timeRemaining = self.timeRemaining - 1
10             if self.timeRemaining <= 0:
11                 self.currentTask = None
12
13     def busy(self):
14         if self.currentTask != None:
15             return True
16         else:
17             return False
18
19     def startNext(self, newtask):
20         self.currentTask = newtask
21         self.timeRemaining = newtask.getPages() * 60/self.pagerate
```

Task class implementation

```
import random

class Task:
    def __init__(self, time):
        self.timestamp = time
        self.pages = random.randrange(1, 21)

    def getStamp(self):
        return self.timestamp

    def getPages(self):
        return self.pages

    def waitTime(self, currenttime):
        return currenttime - self.timestamp
```

Simulation และ สุ่มการเกิด task

```
from pythonds.basic.queue import Queue

import random

def simulation(numSeconds, pagesPerMinute):

    labprinter = Printer(pagesPerMinute)
    printQueue = Queue()
    waitingtimes = []

    for currentSecond in range(numSeconds):

        if newPrintTask():
            task = Task(currentSecond)
            printQueue.enqueue(task)

        if (not labprinter.busy()) and (not printQueue.isEmpty()):
            nexttask = printQueue.dequeue()
            waitingtimes.append(nexttask.waitTime(currentSecond))
            labprinter.startNext(nexttask)

        labprinter.tick()

    averageWait=sum(waitingtimes)/len(waitingtimes)
    print("Average Wait %.2f secs %3d tasks remaining."%(averageWait,printQueue.size()))

def newPrintTask():
    num = random.randrange(1,181)
    if num == 180:
        return True
    else:
        return False
```