Function #2

ฉัตรชัย เกษมทวีโชค

Chatchai.kase@ku.th

Recursive Functions

- Recursive Function เป็นชุดคำสั่งที่ฟังก์ชันมีการเรียกใช้งานฟังก์ชันตัวเอง
- โครงสร้างของ recursive function จะประกอบด้วย
  - Recurrence relation รูปแบบความสัมพันธ์ในการเรียกตัวเอง
  - Termination condition เงื่อนไขในการหยุดการเรียกตัวเอง

```
def f (n):
  if n >= 1:
       result = n + f(n-1)
  else:
       result = 1
  return result
```

$$f(n) = \begin{cases} n + f(n-1); n \geq 1 \\ 1 \quad ; n < 1 \end{cases}$$

Recurrence relation

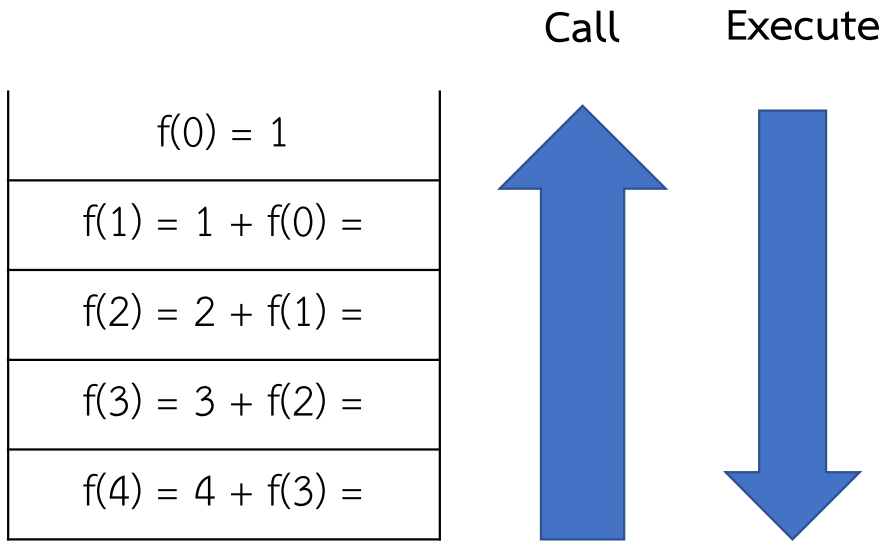
Termination condition

https://www.w3schools.com/python/python_functions.asp

Recursive Functions

- ฟังก์ชันที่เรียกใช้งานก่อนจะต้องรอให้ฟังก์ชันที่จุดสิ้นสุดทำงานได้คำตอบก่อน
- ทำงานย้อนกลับไปฟังก์ชันถัดไปขึ้นมาเรื่อย ๆ จนได้ผลลัพธ์สุดท้าย

```
def f (n):
  if n >= 1:
      result = n + f(n-1)
  else:
      result = 1
  return result
```



https://www.w3schools.com/python/python_functions.asp

Fibonacci sequence

- เป็นเลขอนุกรมที่มีชื่อเสียงในการศึกษาด้านวิทยาการคอมพิวเตอร์ ซึ่งใช้ในการศึกษาด้านอัลกอริทึม
- 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

$$F_n = \begin{cases} F_{n-1} + F_{n-2} & ; n \geq 1 \\ 1 & ; n = 1 \\ 0 & ; n = 0 \end{cases}$$

```
# Function for nth Fibonacci number
 def Fibonacci(n):
    if n<0:
        print("Incorrect input")
    elif n==0:
        return 0
    elif n==1:
        return 1
    else:
        return Fibonacci(n-1)+Fibonacci(n-2)

#This code is contributed by Saket Modi
print(Fibonacci(9))
```

Source : https://www.geeksforgeeks.org/program-for-nth-fibonacci-number/

แบบฝึกหัด 1

- จงเขียนโปรแกรมคำนวณ Factorial number ในรูปแบบ Recursive function และจงทดสอบ 15! มีค่าเท่าไร
- จงเขียนโปรแกรมคำนวณจำนวนนิสิตคงเหลือในรูปแบบ Recurrence relation โดยกำหนดให้ปีที่ 1 รับนิสิตจำนวน 100 คน และมีอัตรานิสิตลาออกเท่ากับ 10% (x = 0.9) จงทดสอบว่า ณ ปีที่ 5 จะเหลือนิสิตประมาณกี่คน

- $f(n) = \begin{cases} n * f(n-1); n > 1 \\ 1 \quad ; n \leq 1 \end{cases}$

- $f(x,n) = \begin{cases} x * f(n-1) & ; n > 1 \\ 100 & ; n = 1 \\ 0 & ; n \leq 0 \end{cases}$

lambda functions

- Anonymous function (ฟังก์ชันไร้ชื่อ หรือนิรนาม) เป็นฟังก์ขันที่ไม่มีการตั้งชื่อ
- โดยทั่วไปฟังก์ชันจะกำหนดชื่อฟังก์ชันเพื่อเรียกใช้งานตามหลังคีย์เวิร์ด def แต่ anonymous function จะใช้คีย์เวิร์ด lambda
- Lambda function จะมีคำสั่งหรือ expression เพียงบรรทัดเดียวเท่านั้น ซึ่งจะมีการประมวลผลคำสั่งและส่งค่าคืนในคำสั่งเดียว

```
lambda arguments: expression
```

```
# Program to show the use of lambda functions
double = lambda x: x * 2


print(double(5))
```

```
def double(x):
    return x * 2
```

lambda functions with condition

```
lambda arguments: true_value if condition else false_value
```

```
data = [5, 3, 8, 7, 1]
check = lambda x: 'Pass' if x>= 5 else 'Fail'


print (check(7))
print ([check(i) for i in data])
```

```
data = [5, 3, 8, 7, 1]
check2 = lambda x: 1 if x >= 5 else 0

print (check2(7))
print ([check2(i) for i in data])
```

แบบฝึกหัด 2

- จงเขียน lambda function ตามคำสั่งต่อไปนี้ โดยกำหนดให้ money = [500, 1000, 300, 800]

ฟังก์ชัน discount รับค่าอาร์กิวเมนท์สองค่า เป็นจำนวนเงิน (x) และเปอร์เซ็นต์ส่วนลด (y) ให้คำนวณหาส่วนลดที่คำนวณได้แล้วส่งผลลัพธ์กลับ
ฟังก์ชัน checknumber รับค่าอาร์กิวเมนท์ (x) ให้ตรวจสอบว่าเป็นค่าตัวเลขหรือไม่ ถ้าใช่ ให้ส่งผลลัพธ์เป็น 1 และถ้าไม่ใช่ ให้ส่งผลลัพธ์เป็น 0
ฟังก์ชัน calDiscount รับค่าอาร์กิวเมนท์ที่ส่งเข้ามากกว่า 500 บาท ให้ส่วนลด 10% และน้อยกว่า 500 บาท ให้ส่วนลด 5% แล้วส่งเป็นผลลัพธ์ส่วนลดกลับ
ฟังก์ชัน calVAT รับค่าจากรายการใน money ให้เพิ่มค่าเป็น 1.07 เท่าของค่าเดิม แล้วส่งเป็นผลลัพธ์กลับ

filter()

- ฟังก์ชัน filter() เป็นฟังก์ชันที่ใช้ผลลัพธ์ของ lambda function เป็นคำตอบที่คัดกรองค่าข้อมูล
- Parameters: `filter(function, sequence)`
  - function เป็นฟังก์ชันตรวจสอบเงื่อนไขของรายการค่าข้อมูล โดยจะส่งค่า return value กลับออกเป็นค่าบูลีน (True หรือ False) หรือค่า 0 และ 1
  - sequence: รายการค่าข้อมูลที่ต้องการคัดเลือกอาจจะเป็น sets, lists, tuples, dictionary (iterable objects)
- Returns:
  - รายการค่าที่ผ่านเงื่อนไขใน function

filter()

- ฟังก์ชัน filter() สามารถใช้ผลลัพธ์ของ lambda function เป็นคำตอบที่เป็นรายการที่มีจำนวนรายการลดลงที่ถูกคัดเลือกรายการแล้ว

my_list	1	5	4	6	8	11	3	12
new_list	4	6	8	12				

```
def vowel(variable):
    letters = ['a', 'e', 'i', 'o', 'u']
    if (variable in letters):
        return True
    else:
        return False


sequence = ['g', 'e', 'u', 'j', 'k', 's', 'p', 'r']
filtered = list(filter(vowel, sequence))
print (filtered)
```

```
my_list = [1, 5, 4, 6, 8, 11, 3, 12]
new_list = list(filter(lambda x: (x%2 == 0) , my_list))

print(new_list)
```

```
my_list2 = [0, 1, 4, 7, 8, 11, 13, 16]
new_list2 = list(filter(lambda x: (x%2 != 0) , my_list2))

print(new_list2)
```

my_list2	0	1	4	7	8	11	13	16
new_list2	1	7	11	13				

แบบฝึกหัด 3

- สร้าง filter ที่สามารถในคำตอบดังต่อไปนี้ เมื่อ ตัวแปร A, B เป็น list และ X เป็น dictionary ที่มีรายการค่าดังต่อไปนี้

  A = [1, 6, 7, 3, 5, 8]

  B = [2, 5, 7, 4, 6, 'A']

  ```
  X = {'Alan': 100, 'Bob': 75, 'Clark': 60, 'David': 55}
  ```

ผลลัพธ์	filter function
C = [6, 7, 8]	
C = [1, 7, 3, 5]	
C = ['A']	
C = ['Alan', 'Bob']	

map()

- ฟังก์ชัน map จะรับค่ารายการใน list, tuple, set มาคำนวณทีละรายการ และส่งผลลัพธ์เป็นรายการค่า

```
map(function, list)
```

```
items = [1, 2, 3, 4, 5]
squared = list(map(lambda x: x**2, items))
```

items	1	2	3	4	5
squared	1	4	9	16	25


```
items = [1, 2, 3, 4, 5]
squared = []
for i in items:
    squared.append(i**2)
```

reduce()

- ฟังก์ชัน reduce จะรับค่ารายการใน list, tuple, set มาคำนวณทีละรายการ และส่งผลลัพธ์เป็นค่าเดียว

```
reduce(function, list)
```

```
from functools import reduce
items = [1, 2, 3, 4, 5]
product = reduce((lambda x, y: x * y), items)
```

items	1	2	3	4	5
product	120				


```
product = 1
list = [1, 2, 3, 4]
for num in list:
    product = product * num
```

แบบฝึกหัด 4

- สร้าง map และ reduce ที่สามารถในคำตอบดังต่อไปนี้ เมื่อ ตัวแปร A, B เป็น list ที่มีรายการค่าดังต่อไปนี้

  A = [1, 2, 3, 4, 5]

  B = [2, 4, 6, 8, 10]

ผลลัพธ์	**map และ reduce function**
C = [15]	
C = [10]	
C = [3, 6, 9, 12, 15]	
C = [4,8]	