2559_2_NumPy_L3

numpy มีคำสั่งให้ใช้มากมาย ขอเสนอคำสั่งที่ลดภาระการเขียน และยังทำงานได้รวดเร็วด้วย ดังตารางข้างล่างนี้ (**4** ฟังก์ชันแรกนี้รับ **x** เป็นอาเรย์หนึ่งมิติ และ **list** ก็ได้ (รับหลายมิติก็ได้ แต่ขอไม่กล่าวถึง)

function	return value
`np.sum(x)`	ผลบวกของทุกค่าใน **x** เช่น `np.sum(np.array([1,1,3,5]))` ได้ 10
`np.prod(x)`	ผลคูณของทุกค่าใน **x** เช่น `np.sum(np.array([1,1,3,5]))` ได้ 15
`np.cumsum(x)` `cumsum` ย่อมาจาก **cumulative sum**	อาเรย์ที่เก็บผลบวกสะสมของค่าใน **x** เช่น `np.cumsum(np.array([1,1,3,5]))` ได้ `array([ 1,  2,  5, 10])`
`np.cumprod(x)` `cumprod` ย่อมาจาก **cumulative product**	อาเรย์ที่เก็บผลคูณสะสมของค่าใน **x** เช่น `np.cumprod(np.array([1,1,3,5]))` ได้ `array([ 1,  1,  3, 15])`

`x[2:4] = 1`
(ถ้า **x** เป็น **list** ทำแบบนี้ไม่ได้)
	ตั้งค่าในอาเรย์ตามช่วงที่กำหนดได้ เช่น
	`x = np.zeros((6,))`
	`x[2:4] = 1` จะได้ **x** มีค่าเป็น
	`array([ 0.,  0.,  1.,  1.,  0.,  0.])`

โจทย์ข้อนี้ให้เขียนฟังก์ชัน **eval_poly(x,coef)** ที่คืนค่าของ **polynomial** $a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$ จากค่า **x** และค่าของสัมประสิทธิ์ $a_0, a_1, \ldots, a_n$ ที่ให้มาในอาเรย์ **coef** เช่น **x = 2, coef = np.array([1,4,10])** ค่าที่คำนวณได้คือ

$$1 + 4\times2 + 10\times2^2 = 49$$

อยากให้นิสิตเขียนคำสั่งในข้อนี้ด้วยคำสั่งของ **numpy** ให้มากสุดเท่าที่จะมากได้ โดย**หลีกเลี่ยงการเขียนคำสั่งวงวน (for, while)** เพื่อฝึกการเลือกใช้คำสั่ง **numpy** ให้เหมาะสม ทำให้ประมวลผลได้รวดเร็ว

```
import numpy as np

def eval_poly(x,coef):
    Y = _____________   # Y เป็น numpy array ขนาดเท่ากับ coef
    _____________       # ทำให้ Y = np.array([1,x,x,...,x]) จำนวนช่องเท่ากับของ coef
    _____________
    Y = _____________   # ทำให้ Y = np.array([1,x,x²,...,xⁿ]) จำนวนช่องเท่ากับของ coef
    return _____________ # Y ทำอะไรบางอย่างกับ coef ก็ได้คำตอบ

exec(input().strip())  # do not remove this line
```

ข้อมูลนำเข้า

คำสั่งในการทดสอบฟังก์ชันที่เขียน

ข้อมูลส่งออก

ผลที่ได้จากคำสั่งที่ป้อนเป็นข้อมูลนำเข้า

ตัวอย่าง

input	output (ทางจอภาพ)
`print(eval_poly(2,np.array([1,4,10])))`	`49.0`