BÁCH KHOA E-LEARNING

Trang của tôi / Khoá học / Học kỳ I năm học 2021-2022 (Semester 1 - Academic year 2021-2022) / Chương Trình Chất Lượng Cao dạy bằng Tiếng Anh (High-Quality training program) / Khoa Khoa học và Kỹ thuật Máy tính (Faculty of Computer Science and Engineering) / Khoa Học Máy Tính / Principles of Programming Languages (CO3005)_Nguyễn Hứa Phùng (CC_HK211) / 5-FP / FP Programming

Đã bắt đầu vào lúc	Tuesday, 14 September 2021, 2:01 PM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Tuesday, 14 September 2021, 3:01 PM
Thời gian thực hiện	59 phút 32 giây
Điểm	9,00/9,00
Điểm	**10,00** của 10,00 (**100**%)

Câu hỏi **1**

Chính xác

Điểm 1,00 của 1,00

Let **lst** be a list of integer and **n** be any value, use **high-order function approach** to write function **dist**(lst,n) that returns the list of pairs of an element of lst and n.

For example:

Test	Result
`dist([1,2,3],4)`	`[(1, 4),(2, 4),(3, 4)]`

Answer: (penalty regime: 0 %)

```
dist = lambda lst,n: map(lambda x: (x,n),lst)
```

	Test	Expected	Got	
✓	`dist([1,2,3],4)`	`[(1, 4),(2, 4),(3, 4)]`	`[(1, 4),(2, 4),(3, 4)]`	✓
✓	`dist([],4)`	`[]`	`[]`	✓
✓	`dist([1,2,3],'a')`	`[(1, 'a'),(2, 'a'),(3, 'a')]`	`[(1, 'a'),(2, 'a'),(3, 'a')]`	✓
✓	`dist([3,4,1,5],6)`	`[(3, 6),(4, 6),(1, 6),(5, 6)]`	`[(3, 6),(4, 6),(1, 6),(5, 6)]`	✓
✓	`dist([1],'a')`	`[(1, 'a')]`	`[(1, 'a')]`	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **2**

Chính xác

Điểm 1,00 của 1,00

Let **lst** be a list of integer and **n** be any value, use **list comprehension approach** to write function **dist**(lst,n) that returns the list of pairs of an element of lst and n.

For example:

Test	Result
`dist([1,2,3],4)`	`[(1, 4),(2, 4),(3, 4)]`

Answer: (penalty regime: 0 %)

```
def dist(lst,n):
    return [(x,n) for x in lst]
```

	Test	Expected	Got	
✓	`dist([1,2,3],4)`	`[(1, 4),(2, 4),(3, 4)]`	`[(1, 4),(2, 4),(3, 4)]`	✓
✓	`dist([],4)`	`[]`	`[]`	✓
✓	`dist([1,2,3],'a')`	`[(1, 'a'),(2, 'a'),(3, 'a')]`	`[(1, 'a'),(2, 'a'),(3, 'a')]`	✓
✓	`dist([3,4,1,5],6)`	`[(3, 6),(4, 6),(1, 6),(5, 6)]`	`[(3, 6),(4, 6),(1, 6),(5, 6)]`	✓
✓	`dist([1],'a')`	`[(1, 'a')]`	`[(1, 'a')]`	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **3**

Chính xác

Điểm 1,00 của 1,00

Let lst be a list of a list of element, use **recursive approach** to write function **flatten**(lst) that returns the list of all elements

For example:

Test	Result
`flatten([[1,2,3],[4,5],[6,7]])`	`[1,2,3,4,5,6,7]`

Answer: (penalty regime: 0 %)

```python
def flatten(lst):
    if not isinstance(lst,list):
        return [lst]

    if len(lst)==0:
        return []

    head = lst[0]
    tail = lst[1:] if len(lst)>1 else []

    return flatten(head) + flatten(tail)
```

	Test	Expected	Got	
✓	`flatten([[1,2,3],[4,5],[6,7]])`	`[1,2,3,4,5,6,7]`	`[1,2,3,4,5,6,7]`	✓
✓	`flatten([[]])`	`[]`	`[]`	✓
✓	`flatten([])`	`[]`	`[]`	✓
✓	`flatten([[1,2,3]])`	`[1,2,3]`	`[1,2,3]`	✓
✓	`flatten([[1],[2],[3],[4],[5,6,7]])`	`[1,2,3,4,5,6,7]`	`[1,2,3,4,5,6,7]`	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **4**

Chính xác

Điểm 1,00 của 1,00

Let lst be a list of a list of element, use **high-order function approach** to write function **flatten**(lst) that returns the list of all elements

For example:

Test	Result
`flatten([[1,2,3],[4,5],[6,7]])`	`[1,2,3,4,5,6,7]`

Answer: (penalty regime: 0 %)

```
from functools import reduce
def flatten(lst):
    return reduce(lambda x,y: x+y,lst,[])
```

	Test	Expected	Got	
✓	`flatten([[1,2,3],[4,5],[6,7]])`	`[1,2,3,4,5,6,7]`	`[1,2,3,4,5,6,7]`	✓
✓	`flatten([[]])`	`[]`	`[]`	✓
✓	`flatten([])`	`[]`	`[]`	✓
✓	`flatten([[1,2,3]])`	`[1,2,3]`	`[1,2,3]`	✓
✓	`flatten([[1],[2],[3],[4],[5,6,7]])`	`[1,2,3,4,5,6,7]`	`[1,2,3,4,5,6,7]`	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **5**

Chính xác

Điểm 1,00 của 1,00

Let lst be a list of a list of element, use **list comprehension approach** to write function **flatten**(lst) that returns the list of all elements

For example:

Test	Result
`flatten([[1,2,3],[4,5],[6,7]])`	`[1,2,3,4,5,6,7]`

Answer: (penalty regime: 0 %)

```python
def flatten(lst):
    return [e for sub in lst for e in sub]
```

	Test	Expected	Got	
✓	`flatten([[1,2,3],[4,5],[6,7]])`	`[1,2,3,4,5,6,7]`	`[1,2,3,4,5,6,7]`	✓
✓	`flatten([[]])`	`[]`	`[]`	✓
✓	`flatten([])`	`[]`	`[]`	✓
✓	`flatten([[1,2,3]])`	`[1,2,3]`	`[1,2,3]`	✓
✓	`flatten([[1],[2],[3],[4],[5,6,7]])`	`[1,2,3,4,5,6,7]`	`[1,2,3,4,5,6,7]`	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **6**

Chính xác

Điểm 1,00 của 1,00

Use recursive approach to write a function lstSquare(n:Int) that returns a list of the squares of the numbers from 1 to n?

For example:

Test	Result
`lstSquare(3)`	`[1,4,9]`

Answer: (penalty regime: 0 %)

```
def lstSquare(n):

    if n==0:
        return []

    return lstSquare(n-1) + [n*n]
```

	Test	Expected	Got	
✓	`lstSquare(3)`	`[1,4,9]`	`[1,4,9]`	✓
✓	`lstSquare(1)`	`[1]`	`[1]`	✓
✓	`lstSquare(5)`	`[1,4,9,16,25]`	`[1,4,9,16,25]`	✓
✓	`lstSquare(4)`	`[1,4,9,16]`	`[1,4,9,16]`	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 7

Chính xác

Điểm 1,00 của 1,00

Use list comprehension approach to write a function lstSquare(n:Int) that returns a list of the squares of the numbers from 1 to n?

For example:

Test	Result
`lstSquare(3)`	`[1,4,9]`

Answer: (penalty regime: 0 %)

```
def lstSquare(n):
    return [n*n for n in range(1,n+1)]
```

	Test	Expected	Got	
✓	`lstSquare(3)`	`[1,4,9]`	`[1,4,9]`	✓
✓	`lstSquare(1)`	`[1]`	`[1]`	✓
✓	`lstSquare(5)`	`[1,4,9,16,25]`	`[1,4,9,16,25]`	✓
✓	`lstSquare(4)`	`[1,4,9,16]`	`[1,4,9,16]`	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **8**

Chính xác

Điểm 1,00 của 1,00

Let **lst** be a list of integer and **n** be an integer, use **recursive approach** to write function **lessThan**(lst,n) that returns the list of all numbers in **lst** less than **n**.

For example:

Test	Result
`lessThan([1,2,3,4,5],4)`	`[1,2,3]`

Answer: (penalty regime: 0 %)

```
def lessThan(lst,n):
    if not isinstance(lst,list):
        return [lst] if lst < n else []

    if len(lst)==0:
        return []

    head = lst[0]
    tail = lst[1:] if len(lst)>1 else []

    return lessThan(head,n) + lessThan(tail,n)
```

	Test	Expected	Got	
✓	`lessThan([1,2,3,4,5],4)`	`[1,2,3]`	`[1,2,3]`	✓
✓	`lessThan([],2)`	`[]`	`[]`	✓
✓	`lessThan([5,2,6,4,1],3)`	`[2,1]`	`[2,1]`	✓
✓	`lessThan([7,6,3,3,5],3)`	`[]`	`[]`	✓
✓	`lessThan([1,2,3,-1,0],6)`	`[1,2,3,-1,0]`	`[1,2,3,-1,0]`	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi **9**

Chính xác

Điểm 1,00 của 1,00

Scala has function compose to compose two functions but Python does not have this function. Write function **compose** that can takes at least two functions as its parameters and returns the composition of these parameter functions. For example **compose(f,g,h)(x)** is defined as **f(g(h(x)))**.

For example:

Test	Result

`f = compose(increase,square)`
`print(f(3)) #increase(square(3)) = 10`
	`10`

Answer: (penalty regime: 0 %)

```python
from functools import reduce

increase = lambda x: x+1
square = lambda x: x*x

def compose(f1,f2, *args):

    funcs = [f1] + [f2] + list(args)
    def f(x):
        return reduce(lambda f,g: g(f), funcs[::-1] ,x)

    return f

```

	Test	Expected	Got	

✓	`f = compose(increase,square)`
	`print(f(3)) #increase(square(3)) = 10`
		`10`	`10`	✓

✓	`f = compose(increase,square,double)`
	`print(f(3))`
		`37`	`37`	✓

✓	`f = compose(increase,square,double,decrease)`
	`print(f(3))`
		`17`	`17`	✓

✓	`try:`
	`    f = compose(increase)`
	`except TypeError:`
	`    print("compose() missing 1 required positional argument")`
		`compose() missing 1 required positional argument`	`compose() missing 1 required positional argument`	✓

✓	`try:`
	`    f = compose()`
	`except TypeError:`
	`    print("compose() missing 1 required positional argument")`
		`compose() missing 1 required positional argument`	`compose() missing 1 required positional argument`	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

◄ FP Quiz

Chuyển tới...

Link Video of session 14/09/2021 ►