BÁCH KHOA E-LEARNING

Đã bắt đầu vào lúc	Tuesday, 14 September 2021, 8:11 AM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Tuesday, 14 September 2021, 10:22 AM
Thời gian thực hiện	2 giờ 10 phút
Điểm	7,00/7,00
Điểm	**10,00** của 10,00 (**100**%)

Câu hỏi **1**

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(size):
    if (size > 0):
        return lstSquare(size-1)+[size*size]
    else:
        return []
```

	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 **2**

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 square(n):
    return n*n


def lstSquare(size):
    result = []
    result = [square(number) for number in range(1, size+1)]
    return result

```

	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 **3**

Chính xác

Điểm 1,00 của 1,00

Use high-order function approach to write function lstSquare(n:Int) to return a list of i square for i from 1 to n?

For example:

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

Answer: (penalty regime: 0 %)

```
def lstSquare(size):
    return list(map(lambda x: x * x, list(range(1, size+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 **4**

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 %)

```
def dist(lst, n):
    return list(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 **5**

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 ulti(a, b):
    return (a, b)


def dist(lst, n):
    return list([ulti(number, n) for number 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 **6**

Chính xác

Điểm 1,00 của 1,00

Let **lst** be a list of integer and **n** be any value, use **recursive 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):
    if len(lst):
        return [(lst[0], n)]+dist(lst[1:], n)
    else:
        return []
```

	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 **7**

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 %)

```
def compose (*functions):
    if len(functions) < 2: raise TypeError

    def inner(arg):
        for f in reversed(functions):
            arg = f(arg)
        return arg
    return inner
```

	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 của buổi học 14/09/2021 ►

Copyright 2007-2021 Trường Đại Học Bách Khoa - ĐHQG Tp.HCM. All Rights Reserved.
Địa chỉ: Nhà A1- 268 Lý Thường Kiệt, Phường 14, Quận 10, Tp.HCM.
Email: elearning@hcmut.edu.vn
Phát triển dựa trên hệ thống Moodle