Đã bắt đầu vào lúc	Monday, 14 February 2022, 9:55 PM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Monday, 14 February 2022, 10:52 PM
Thời gian thực hiện	57 phút 28 giây
Điểm	9,10 của 10,00 (91%)

Câu hỏi 1

Chính xác

Điểm 1,00 của 1,00

Let **lst** be a list of integer and **n** be an integer, use *list comprehension 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: 10, 20, ... %)

```
def lessThan(lst,n):
    return [x for x in lst if x < n]
```

	Test	Expected	Got	
✔	`lessThan([1,2,3,4,5],4)`	`[1,2,3]`	`[1,2,3]`	✔
✔	`lessThan([],4)`	`[]`	`[]`	✔
✔	`lessThan([5,2,6,4,1],4)`	`[2,1]`	`[2,1]`	✔
✔	`lessThan([7,6,4,4,5],4)`	`[]`	`[]`	✔
✔	`lessThan([1,2,3,-1,0],4)`	`[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 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: 10, 20, ... %)

```
def lstSquare(n):
    return [x*x for x 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 3

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: 10, 20, ... %)

```
def lstSquare(n):
    if n == 0:
        return []
    else:
        return lstSquare(n-1) + [n**2]
```

	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 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: 10, 20, ... %)*

```
def flatten(lst):
    if lst == []:
        return lst
    if isinstance(lst[0], list):
        return flatten(lst[0]) + flatten(lst[1:])
    return lst[:1] + flatten(lst[1:])
```

	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 0,60 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: 10, 20, ... %)

```
def dist(lst,n):
    if lst == []:
        return []
    return [(lst[0],n)] + dist(lst[1:],n)
```

	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. Tính toán cho lần làm bài trước đó, điểm 0,60/1,00.

Câu hỏi 6

Chính xác

Điểm 0,90 của 1,00

Let **lst** be a list of integer and **n** be an integer, use *high-order function 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: 10, 20, ... %)

```
def lessThan(lst,n):
    return list(filter(lambda x: x < n, lst))
```

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

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00. Tính toán cho lần làm bài trước đó, điểm 0,90/1,00.

Câu hỏi 7

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: 10, 20, ... %)

```
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 8

Chính xác

Điểm 0,60 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: 10, 20, ... %)*

```
from functools import reduce
def flatten(lst):
    return [] if lst == [] else list(reduce(lambda prev,curr: prev + curr, 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. Tính toán cho lần làm bài trước đó, điểm 0,60/1,00.

Câu hỏi 9

Chính xác

Điểm 1,00 của 1,00

Cho biết Python dùng phép toán ** để tính luỹ thừa, ví dụ 2**3 sẽ có kết quả là 8. Hãy viết hàm powGen(x) có chức năng như một closure để tạo ra các hàm luỹ thừa ứng với thông số mà nó nhận được. Ví dụ powGen(2) sẽ trả về hàm luỹ thừa hai, powGen(3) trả về hàm luỹ thừa ba,...

For example:

Test	*Result*
`square = powGen(2)` `print(square(4))`	`16`

Answer: *(penalty regime: 10, 20, ... %)*

```
def powGen(x):
    def exp(y):
        return y**x
    return exp
```

	Test	*Expected*	*Got*	
✔	`square = powGen(2)` `print(square(4))`	`16`	`16`	✔
✔	`cube = powGen(3)` `print(cube(2))`	`8`	`8`	✔
✔	`iden=powGen(1)` `print(iden(3))`	`3`	`3`	✔
✔	`one = powGen(0)` `print(one(3))`	`1`	`1`	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 10

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 %)*

```
from functools import reduce
def compose(*func):
    def inner(f,g):
        return lambda x: f(g(x))
    return reduce(inner,func,lambda x:x)
```

	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`	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

◄ Case Study: Python

Chuyển tới...

Slides: AST ►