BY: Zhu_Ming

<CS1010S>

Tutorial 5

Working with Sequences : Tuples

BY: Zhu_Ming

<CS1010S>

How’s the Midterm??

Lecture Recap

1. Tuples
2. Box & Pointer

1. Tuples

- A immutable sequence of elements

```
tup = (1,2,3)
tup[2] = 0 # will raise TypeError
```

- Is a reference type
- Element could be any type
- Round brackets *e.g. (1, 2)*
- COMMA matters ,,,,

```
non_tup = (1) # Wrong, non_tup = 1
tup = (1,) # Correct
```

Important

Code tracing is important!!

Box & Pointer is a tool to trace reference type object!!

Recap!!

Primitive Type: (int, str, float, bool, none)
- fundamental data structure that predefined
- SAME identity!!

```
a = “same”
b = “same”
a == b # True
a is b # True
```

Reference Type:
- Look alike ⇍/⇒ Same Identity
- Same Identity ⇒ Look alike

```
tup1 = (1,2)
tup2 = (1,2)
tup1 == tup2 # True
tup1 is tup2 # False
```

- ## Tuple Addition

```
seq = (1, 2)
seq += (3, )
(1, 2) + (3, 4) vs (1, 2) + ((3, 4), )
(1, 2, 3, 4)              (1, 2, (3, 4))
```

Concatenation

Always remember tuple is immutable!!

There is NO “update” of tuple!

Create an **entirely new tuple** when trying to “update” it

As a result, O(n) time & space

- ## Looping a Tuple

```
tup = (1, 2, 3)
for ele in tup:
    print(ele)

>>> 1
>>> 2
>>> 3
```

PythonTips!

```
for idx in range(len(tup)):
    print(i)
    print(tup[i])

for idx, ele in enumerate(tup):
    print(idx)
    print(ele)
```

- ## Useful Function

```
- len(tup)        # returns length of tuple
- map(fn, seq)    # applies fn to every element in seq
- filter(fn, seq) # keeps elements in sq where fn returns True

**Notice that map & filter return an Object

tup = (1,1)
map(lambda x: x * 2, tup)           # <map object at 0x000001CB1098EBF0>
tuple(map(lambda x: x * 2, tup))    # (2, 2)
```

PythonTips!

`map` & `filter` is like a for loop.

`map` & `filter` make code cleaner.

2. Box & Pointer Diagram

- A way to depicting the memory location of variables
- A visual representation of how computer store the variables

PythonTips!

Pythontutor is expert in Box & Pointer Diagram

BY: Zhu_Ming

Any Questions?

Question_1.tut05

Draw box-and-pointer diagrams for the values of the following tuples:

```
tup1 = ((1, 2, (3,)), (4, 5), (6, 7))
tup2 = (1, (2, 3, (4,)))
tup3 = (1, (2, (3, (4, 5))))
```

Question_2.tut05

Write expressions using index notation that will return the value 1 when the identifier tup is bound to the following values:

```
(7, (6, 5, 4), 3, (2, 1))                  # tup[3][1]
((7), (6, 5, 4), (3, 2), 1)                # tup[3]
(7, (6, ), (5, (4, )), (3, (2, (1, ))))   # tup[3][1][1][0]
(7, ((6, 5), (4, ), 3, 2), ((1, ), ))     # tup[2][0][0]
```

Question_3.tut05

Write a function odd_indices that takes in a tuple as its only argument and returns a tuple containing all the elements with odd indices from the input tuple.

```
def odd_indices(tup):
    return tup[1::2]
```

```
def odd_indices(tup):
    new_tup = ()
    for idx, ele in enumerate(tup):
        if idx % 2:
            new_tup += (ele)
    return new_tup
```

Question_3.tut05

Write a function odd_indices that takes in a tuple as its only argument and returns a tuple containing all the elements with odd indices from the input tuple.

```
def odd_indices(tup):
    return tup[1::2]


def odd_indices(tup):
    new_tup = ()
    for idx, ele in enumerate(tup):
        if idx % 2:
            new_tup += (ele,)
    return new_tup

def odd_indices(tup):
    return  tuple(tup[i] for i in range(len(tup)) if i % 2 == 1)

def odd_indices(tup):
    if len(tup) < 2:
        return ()
    else:
        return (tup[1], ) + odd_indices(tup[2:])
```

Question_4.tut05

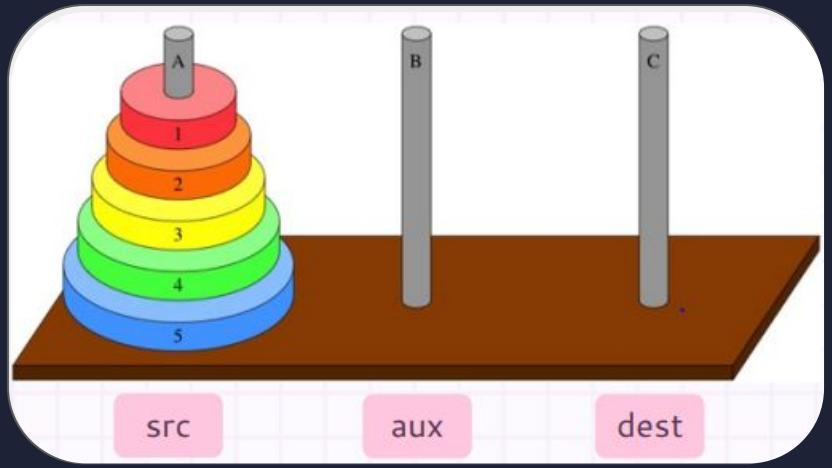
Write a function even_odd_sums that takes in a tuple of numbers and returns a tuple of two elements
(sum(even-indexed elements), sum(odd-indexed elements))

```
def even_indices(tup):
    return tup[0::2]

def even_odd_sums(tup):
    return (sum(even_indices(tup)), sum(odd_indices(tup))
```

Question_5.tut05

Write a function hanoi that returns a tuple of disk moves to solve the Tower of Hanoi game




```
def hanoi(num, src, dest, aux):
    if num == 0:
        return ()
    if num == 1:
        return ((src, dest), )
    return hanoi(num - 1, src, aux, dest) +
           ((src, dest), ) +
           hanoi(num - 1, aux, dest, src)
```

1. move **(n - 1)** from **src** to **aux**
2. move **(1)** from **src** to **dest**
3. move **(n - 1)** from **aux** to **dest**

Thank You!!

The End

See you next lesson

Extra_Question.tut05

EXTRA Practices

EXTRA_Question.tut05

(EXTRA)

who Am i, where Am i

Question 1

```
a = (1, 2)
b = 2
c = (a, b)

print(a is c[0])
print((1, 2) in c)
print((1) in c)
```

EXTRA_Question.tut05

OUTPUT:

```
True
True
False
```

EXTRA_Question.tut05

(EXTRA)

Butter & Margarine

QUESTION 2

```
a = (1, 2)
b = (1, 2, 3)
c = a + b

print(b in c[2:])
print(b is c[2:])
print(b == c[2:])
```

EXTRA_Question.tut05

```
False
False
True
```

EXTRA_Question.tut05

(EXTRA)

I will remember you

QUESTION 2

```
a = (1, 2)
b = (a, 3)
a = (1, 2, 3)

print(a in b[0])
```

EXTRA_Question.tut05

False

BY: Zhu_Ming

Any Questions?