CSC148 Notes

Tianyu Du

January 10, 2018

Contents

1 Lecture 1. Jan. 10 2017

Outlines

1. Construct solutions to real world problems.
2. Abstract data types.
3. Recursion.
4. Exceptions.
5. Design.
6. Efficiency.

1.1 Object

```
>>> s1 = 'word'
>>> s2 = 'sword'[1:]
>>> s1 == s2
True
>>> s1 is s2 # s1 and s2 are different objects.
False
```

```
>>> n1 = 255
>>> n2 = 255
>>> n1 == n2
True
>>> n1 is n2
True
>>> n3 = 257
>>> n4 = 257
>>> n3 is n4
False
```

Object Components of object:

- Identifier.
- Type.
- Value.

1.2 Review function design recipe

***Repeated* function** Check list:

1. Header.
2. Type contract.
3. Doc string.
4. Function body.
5. Test.

Design recipe

```
from typing import List

def repeated(word: str, n: int) -> List[str]:
'''
Repeated - return a list of word n times.

>>> repeated('a', 2)
['a', 'a']

>>> repeated('a', 0)
[]
'''
return [word] * n
```

1.3 Point

```
class Point:
’’’
Represent a two-dimensional point.

x - horizontal position.
y - vertical position.
’’’

x: float
y: float

def __init__(self, x, y) -> None:
’’’

’’’
self.x, self.y = x, y

def distance_to_origin(self):
return (self.x **2 + self.y ** 2) ** .5

def __eq__(self, other: Any) -> bool:
’’’
Return whether self is equivalent to other
>>> Point(3, 5) == Point(3.0, 5.0)
True
>>> Point(3, 5) == Point(5, 3)
False
>>> Point(3, 5) == 7
False
’’’
return (type(self) == type(other)
and self.x == other.x
and self.y == other.y)

def __str__(self) -> str:
’’’
Return a string representing this point itself.

>>> print(Point(3, 5))
(3.0, 5.0)
’’’
return ’’({}, {})’’.format(self.x, self.y)
```

1.4 Build API

Define a class API:

1. Choose a class name and write a brief description in the class doc string.

   ```
   Point
   ```

2. Write some examples of client code that uses you class.

   ```
   p = Point(3, 4)
   p.distance_to_origin()
   ```