Python 101

exam02

Announcements

exam: exam02 13 Nov 8-9pm lec06-13 + Quiz and Homework MCQ, short questions, coding

CompE @ LTE 102/103

ME @ LTE 201/202

EE and CE @ LTW 102/103

Recap

Python L06 - L13

A. `L06 - FILE: open, read, write, close`

.read(), .readlines(), .write(), .close()
.split(), .join()
path with ../ and ./,
break, continue, zip, enumerate

Python L06 - L13

```
myfile = open( 'numbers.txt' ) or open( 'numbers.txt', "r" )
myfile = open( 'numbers.txt', "w" )
```

Python L06 - L13

myfile = open('numbers.txt') or open('numbers.txt', "r")

myfile = open('numbers.txt', "w")

myfile.read() gives what? myfile.readlines() gives what?

Python L06 - L13

myfile = open('numbers.txt') or open('numbers.txt', "r")

myfile = open('numbers.txt', "w")

myfile.read() gives what? myfile.readlines() gives what?

```
my_string = " ZJUI is more selfish "
a = my_string.split( '  ' ) or .split( )
or .split( 'a' ) or .split('\n')
```

Python L06 - L13

myfile = open('numbers.txt') or open('numbers.txt', "r")

myfile = open('numbers.txt', "w")

myfile.read() gives what? myfile.readlines() gives what?

```
my_string = " ZJUI is more selfish "
a = my_string.split( '  ' ) or .split( )
or .split( 'a' ) or .split('\n')


my_list = [ 'All','handsome','pretty']
c = ' '.join( my_list ) or 'others'.join( my_lis
```

Python L06 - L13

myfile = open('numbers.txt') or open('numbers.txt', "r")

myfile = open('numbers.txt', "w")

myfile.read() gives what? myfile.readlines() gives what?

```
my_string = " ZJUI is more selfish "
a = my_string.split( '  ' ) or .split( )
or .split( 'a' ) or .split('\n')


my_list = [ 'All','handsome','pretty']
c = ' '.join( my_list ) or 'others'.join( my_lis
```

break, continue, zip, enumerate

```
qs = [ 'name', 'quest', 'favourite' ]
ps = [ 'Meimei', 'Have fun', 'Fun' ]
```

for q, a in zip(qs,ps):

for i, j in enumerate(qs):

Python L06 - L13

A. L07 - Mutable vs Immutable

list? dict? tuple? str? int? float? bool? and others?
mutable can change without return from function
is, id

```
def appender( q ):
    aList.append( 3 )

aList = [ ]
appender( )
print( aList )
```

Python L06 - L13

A. L08 - Lists in list or 2D list using [a][b] indexing, dictionary {}

```
a = [[1,2],[4,5]]
```

```
d = {1:'1', Two':2, 3.0:3.0 4:'Four'}
```

or

```
model = { }
```

a key of dictionary has to be immutable

```
model['iPhone XS'] = 'Apple'
```

indexing using [ ], d.items(), d.keys(), d.values( )
Dictionary sorting by value and key
Applications (Important!): encoding, decoding, counter

Python L06 - L13

A. `L09 - Input Output`

input, process, output
dictreader
requests.get, .text

Python L06 - L13

A. `L09 - Input Output`

input, process, output
dictreader
requests.get, .text

B. `L10 - Numpy`

i. import numpy
ii. numpy.array vs list, numpy vs math
iii. `1D x = np.array([1, 2...])` and `2D x = np.array([[11,12...], ... [21,22...]])`
iv. numpy.zeros(x,y), numpy.ones(x,y)
v. x.shape, x.dtype, x*x, x>n,
vi. numpy.sin(x), numpy.exp(x) and others
vii. np.array is mutable
viii. x.sort(i) 0=column, 1,nothing=row, x.tolist(), x.argsort()
ix. np.linspace(start, finish, n)

Python L06 - L13

A. `L11 - matplotlib`

import matplotlib.pyplot as plt
plt.plot(x,y, ...),
colors and markers: ’r’,’g’,’b’,’k’,’y’,’-’,’–’,’o’,’x’,
plt.show()
x.lim, ylim, xticks, yticks, xlabel,
ylabel, legend, label, title, savefig

Python L06 - L13

A. `L11 - matplotlib`

import matplotlib.pyplot as plt
plt.plot(x,y, ...),
colors and markers: ’r’,’g’,’b’,’k’,’y’,’-’,’–’,’o’,’x’,
plt.show()
x.lim, ylim, xticks, yticks, xlabel,
ylabel, legend, label, title, savefig

B. `L12 - State`

Modeling - Analytical, Numerical
Forward and backward difference, Monitor state

Python L06 - L13

A. L13 - Random

import numpy.random as npr
npr.uniform() or (x,y,size=?),
npr.randint() or (x,y,size=?),
npr.normal() or (size=?)*y+x
plt.hist(x, bins=?)
npr.choice(), npr.shuffle()
Idea behind Monte Carlo Integration

```
x = np.random.randint(0,100,size=(10000,1))
plt.hist(x)
plt.show()
```

Python L06 - L13

A. L13 - Random

import numpy.random as npr
npr.uniform() or (x,y,size=?),
npr.randint() or (x,y,size=?),
npr.normal() or (size=?)*y+x
plt.hist(x, bins=?)
npr.choice(), npr.shuffle()
Idea behind Monte Carlo Integration

```
x = np.random.randint(0,100,size=(10000,1))
plt.hist(x)
plt.show()


x = np.array( range( 1,53 ) )
y = np.random.shuffle(x)
```