CS Final-Sylvia杜竞耘 (Q3 and Q6 added)

2. The statement ‘6\*(1/3)’means 6 multiplies 1/3, while \* means multiply. Python calculates the parameter first and gets a result in decimals. Then it calculates 6 multiplies the decimal and get a decimal 2.0 .

3.

def loan () :

amount = input ('enter amount\n')

term = input ('enter term\n')

r = 3

print ('Loan Amount: $' + amount + ' Term: ' + term + ' years')

print ('Interest Rate Monthly Payment')

term = int (term)

while 3 <= r <= 18 :

rate = r / 1200

n = 12 \* term

d = ((1 + rate) \*\* n - 1) / rate \* (1 + rate) \*\* n

r = str (r)

d = str (d)

print (r + '% ' + d)

r = int (r)

r += 1

###test

>>> loan()

enter amount

350000

enter term

30

Loan Amount: $350000 Term: 30 years

Interest Rate Monthly Payment

3% 1431.6925762765222

4% 2299.731323459162

5% 3718.3187859186737

6% 6049.767395020314

7% 9901.891507227487

8% 16298.168023463886

9% 26967.906239153155

10% 44842.20174270274

11% 74903.38679616967

12% 125642.70702615149

13% 211566.06291851753

14% 357508.15477942646

15% 606070.7867366641

16% 1030464.3878218869

17% 1756703.7660905363

18% 3002012.9749854123

>>>

4. a and b are correct. a is a list of three integers and a string. B is a list of two integers and a nestedlist with two integers. C is not syntactically correct because the two items in the list should be divided by ‘,’.

5.

a)

a = list (input ('enter product code\n'))

if 'Z' in a and '9' in a :

print ('Verified')

else :

print ('Failed')

###test1

enter product code

Za9

Verified

###test2

enter product code

sa9

Failed

b)

a = list (input ('enter product code\n'))

b = len (a) - 3

c = a[b:]

d = ''

f = d.join(c)

print (f)

###test

enter product code

asdgf

dgf

6.

encode = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']

code = ['.\_', '\_...', '\_.\_.', '\_..', '.', '..\_.', '\_\_.', '....', '..', '.\_\_\_', '\_.\_', '.\_..', '\_\_', '\_.', '\_\_\_', '.\_\_.', '\_\_.\_', '.\_.', '...', '\_', '..\_', '...\_', '.\_\_', '\_..\_', '\_.\_\_', '\_\_..']

def morse () :

c = dict (zip (encode, code))

a = input ('enter the sentence\n')

for key in a :

if key == ' ' :

print ('')

elif key == '.' :

print ('')

else :

print (c[key])

def decrypt () :

c = dict (zip (code, encode))

while True :

a = input ('enter the message or enter done to exit\n')

if a == 'done' :

return

else :

print (c[a])

def main () :

a = input ('code or decode?\n')

if a == 'code' :

morse ()

elif a == 'decode' :

decrypt ()

else :

print ('ERROR')

main ()

###test1

code or decode?

code

enter the sentence

HELLO WORLD

....

.

.\_..

.\_..

\_\_\_

.\_\_

\_\_\_

.\_.

.\_..

\_..

>>>

###test2

code or decode?

decode

enter the message or enter done to exit

....

H

enter the message or enter done to exit

.

E

enter the message or enter done to exit

done

>>>

7.a)[somefunction, def, print, return], [(n1, n2)]

b)[n1, n2, d, “ohehe”, num1]

c) [a, b, c, d, num1]

d)print only output the result while return can save the result.

e)local scope is the scope in the function defined, while the gobal scope is the space in the ‘shell’.

8. not finished

x1y1 = '---'

x2y1 = '---'

x3y1 = '---'

x1y2 = '---'

x2y2 = '---'

x3y2 = '---'

x1y3 = '---'

x2y3 = '---'

x3y3 = '---'

print (x1y1, x2y1, x3y1)

print (x1y2, x2y2, x3y2)

print (x1y3, x2y3, x3y3)

p1 = input ('enter p1 move\n')

if p1 == 'x1y1':

x1y1 = ' X '

elif p1 == 'x2y1':

x2y1 = ' X '

elif p1 == 'x3y1':

x3y1 = ' X '

elif p1 == 'x1y2':

x1y2 = ' X '

elif p1 == 'x2y2':

x2y2 = ' X '

elif p1 == 'x3y2':

x3y2 = ' X '

elif p1 == 'x1y3':

x1y3 = ' X '

elif p1 == 'x2y3':

x2y3 = ' X '

elif p1 == 'x3y3':

x3y3 = ' X '

print (x1y1, x2y1, x3y1)

print (x1y2, x2y2, x3y2)

print (x1y3, x2y3, x3y3)

p2 = input ('enter p2 move\n')

if p2 == p1 :

print ('Invalid move')

elif p2 == 'x1y1':

x1y1 = ' O '

elif p2 == 'x2y1':

x2y1 = ' O '

elif p2 == 'x3y1':

x3y1 = ' O '

elif p2 == 'x1y2':

x1y2 = ' O '

elif p2 == 'x2y2':

x2y2 = ' O '

elif p2 == 'x3y2':

x3y2 = ' O '

elif p2 == 'x1y3':

x1y3 = ' O '

elif p2 == 'x2y3':

x2y3 = ' O '

elif p2 == 'x3y3':

x3y3 = ' O '

print (x1y1, x2y1, x3y1)

print (x1y2, x2y2, x3y2)

print (x1y3, x2y3, x3y3)

9.

a = list (input ('enter 6-digit color code\n'))

r = a[:2]

b = a[2:4]

g = a[4:]

d = {'A':10, 'B':11, 'C':12, 'D':13, 'E':14, 'F':15}

r1 = d[r[0]] \* 16 + d[r[1]]

b1 = int(b[0]) \* 16 + int(b[1])

g1 = int(g[0]) \* 16 + int(g[1])

print (r1, b1, g1)

###test

enter 6-digit color code

FF6347

255 99 71

10.

def mes () :

try:

fin = open ('fin.txt', 'r')

fout = open ('fout.txt', 'w')

d = {'A':'F', 'B':'G', 'C':'L', 'D':'R', 'E':'P', 'H':'J', 'I':'K'}

for line in fin :

for a in line:

line.replace(a, d[a])

fout.write (line)

fin.close ()

fout.close ()

except :

print ('something wrong')

11.

a)

def printedTime (s) :

h = str (int (s/3600))

m = str (int ((s%3600) / 60))

s = str ((s%3600) % 60)

print (h + ':' + m + ':' + s)

###test

printedTime (3600)

1:0:0

b)

def isAfter (sa, sb) :

if sa > sb :

return True

else :

return False