# Coding habit

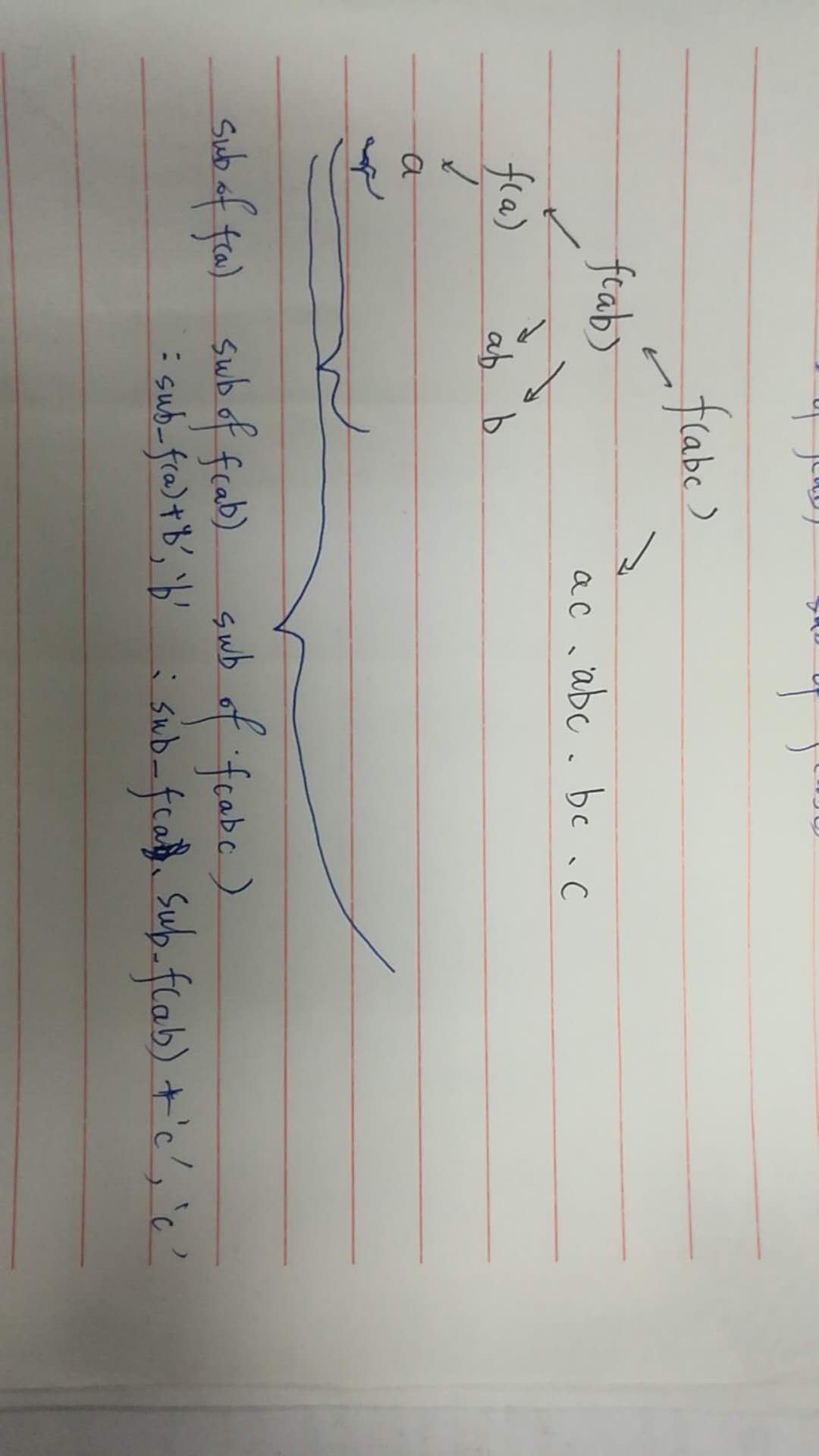
**1, how to analyze a program.**

From main function first, draw the function tree.

# 一些坑与相关解决

**1, : find all the combinations of string, e.g. given ‘abc’, the solution is [‘a’,’b’,’c’,’ab’,’ac’,’bc’,’abc’]**

 "maximum recursion depth exceeded in comparison"

It is a guard against a stack overflow, yes. Python (or rather, the CPython implementation) doesn't optimize tail recursion, and unbridled recursion causes stack overflows. You can change the recursion limit with [sys.setrecursionlimit](http://docs.python.org/library/sys.html#sys.setrecursionlimit), but doing so is dangerous -- the standard limit is a little conservative, but Python stackframes can be quite big.

Python isn't a functional language and tail recursion is not a particularly efficient technique. Rewriting the algorithm iteratively, if possible, is generally a better idea.

Solution : visualized graphic is following.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | | https://github.com/sebrenner/Mit-6.00-OCW-Problem-Set-Solutions/blob/master/ps6.py#L399 | | def build\_substrings(string): | |  | """ | |  | Works on the premiss that given a set of the substrings of a string the | |  | the subsets of a string with one more char is the formed by taking all the | |  | substrings in the known subset and also adding to them the set formed by | |  | adding the character to every element in the old set and then adding the | |  | new char. | |  |  | |  | """ | |  | result = [] | |  | if len(string) == 1: | |  | result.append(string) | |  | else: | |  | for substring in build\_substrings(string[:-1]): | |  | result.append(substring) | |  | substring = substring + string[-1] | |  | result.append(substring) | |  | result.append(string[-1]) | |  | result = list(set(result)) # Convert result into a set. Sets have no duplicates. Then convert back to list. | |  | result.sort() | |  | # now iterate through substrings and sort the characters of each substring | |  | #for each in | |  | return result | |  |  | |

**2, something not works in using python version but does in old version**

**for example:**

<https://docs.python.org/3/whatsnew/3.0.html>

|  |
| --- |
| [dict](https://docs.python.org/3/library/stdtypes.html#dict) methods [dict.keys()](https://docs.python.org/3/library/stdtypes.html#dict.keys), [dict.items()](https://docs.python.org/3/library/stdtypes.html#dict.items) and [dict.values()](https://docs.python.org/3/library/stdtypes.html#dict.values) return “views” instead of lists. For example, this no longer works: k = d.keys(); k.sort(). Use k = sorted(d) instead (this works in Python 2.5 too and is just as efficient).  MORE explaination:  as I understand it a "view" is an iterator, and an iterator does not have the sort function. Change it to  sorted(all\_colors)  according to the documentation |

**for instance again:**

<https://stackoverflow.com/questions/35900265/how-do-i-fix-dict-values-indexing-for-python-3x?lq=1>

|  |
| --- |
| **'dict\_values' object does not support indexing**  In Python 2.x, dict.values returns a list. This was changed in Python 3.x to return a view. You can simply convert it back to a list to get the old behavior:  >>> numbers\_dict = {1:66, 2:2, 3:3, 4:4, 5:5}  >>> print ("First Item: %s" % list(numbers\_dict.values())[0])  First Item: 66  Note that this is not really the "First" item, as dicts are inherently unordered collections. |

# for novice

|  |  |
| --- | --- |
| reason | frequency |
| indentation |  |
| index |  |
| syntax |  |
| reset the state |  |
| condition |  |
| method () missing |  |

pylab.plot(x,y) len(x)!=len(y)

**pop and index**

let’s assume that we have following snippets of code, run it separately. and the 2nd snippet would raise error: index out of range, because with the operation ‘pop’, the length would reduce, but the number of loop won’t change, which beyond the length of modified seasons list. Here give three solutions to this: **use enumerate[good]; reverse the list, and iterate it reversely[GOOD];**use while and change loop index i enumarately.

|  |
| --- |
| seasons = ['Spring', 'Summer', 'Fall', 'Winter']  for k,v in enumerate(seasons):  print(k)  if k==2:seasons.pop(k);print(seasons[k])  else:print(seasons[k])  ##########################  seasons = ['Spring', 'Summer', 'Fall', 'Winter']  for k in range(len(seasons)):  print(k)  if k==2:seasons.pop(k);print(seasons[k])  else:print(seasons[k]) |

|  |
| --- |
| source code  # i=0  # while i <len(self.viruses):  # print(self.getTotalPop(),i)  # if self.viruses[i].doesClear():  # self.viruses.pop(i)  # else: i+=1  # solution2:  # for index, virus in reversed( list( enumerate( self.viruses ) ) ):  # if virus.doesClear():  # # print( "Virus clears")  # # pop virus from viruses list  # self.viruses.pop( index )  # solution3:  for k,v in enumerate(self.viruses):  if v.doesClear():  self.viruses.pop(k) |

**Note the indentation: some** statements inside or outside of “if ” or loop might lead a mistake. OR the indentation of some lines of code in the end of a block of code might easily be omitted.

**be careful about the number,** e.g. find the combinations of base(a,b,c) that satisfy equation: 6a+9b+20c=n

coef\_a=int(n/6)+1 # n/6 is the number used to iteratively generate

coef\_b=int(n/9)+1 # all feasible coefficients of a,b,c, for example

coef\_c=int(n/20)+13

**python’s index:**

* 1. [start, end], generally speaking, from start to end-1, e.g. slicing, str.find(sub[,start[,end]]) and so on
  2. [ps2a\_selling] index () out of range because the preceding command ‘coef\_a=n/6’ not the correct one ‘coef\_a=n/6+1’
  3. [ps3a] mix the inner loop index and the outside
  4. [ps3a] Forget to plus the base(missing 'i+'), the previous version is 'target[i:lenkey] while executing a loop

**indentation:**

For loop:【return】statement is in the 【for】block of code, so the loop won’t work for enough times

**syntax:**

if a line of correct code cause invalid syntax, the reason may be the syntax error caused by the last line of code. e.g.

print(‘dfsdfsdfds}’)

**Reset the state sign**

|  |
| --- |
| def getResistPop(self, drugResist):  """  Get the population of virus particles resistant to the drugs listed in  drugResist.  drugResist: Which drug resistances to include in the population (a list  of strings - e.g. ['guttagonol'] or ['guttagonol', 'grimpex'])  returns: the population of viruses (an integer) with resistances to all  drugs in the drugResist list.  """  # TODO  popResisV=0  for virus in self.viruses:  **isResist=True # this should be reset in every check of the virus**  # if type(virus)==ResistantVirus:  # ??? is it right  # the instance of superclass would not have the attribute of  # its subclass, but vice-versa, i.e. if a statement is instance.attribute,  # then it will directly override that of superclass, no matter how many arguments of the method  for drug in drugResist:  isResist=isResist and virus.getResistance(drug)  if isResist: popResisV+=1  print(isResist,popResisV)  return popResisV |

**Conditi-on is important:**

critical point : wrong critical point might lead to one more loop

if maxWork<=0:

break

else:

best[betName]=subjects[betName]

maxWork-=subjects[betName][WORK]

**forget the() for the method of object**

ipython, cPython,

当我需要分析代码并优化代码时，在spyder中的ipython console中输入以下代码时，提示出错，原因是该代码是在ipython 中运行的，实际应该在shell中运行，即cmd，打开cmd后，输入 // RIGHT CODE 即可

|  |
| --- |
| **// SOURCE CODE**  //  python -m cProfile -s time circuit.py<tests/5devadas13.in  File "<ipython-input-22-1ea890356c30>", line 1  python -m cProfile -s time circuit.py<tests/5devadas13.in  ^  SyntaxError: invalid syntax |
| **// WRONG CODE**  //  python -m cProfile -s time circuit.py<tests/5devadas13.in |
| **// RIGHT CODE**  //  A:\001SimulationLab\pylab\6.006\ps2\circuit>python -m cProfile -s time circuit.py < tests/5devadas13.in |
|  |

**file.readline()**

情况：读文件读不了第一行？？

原因是file.readline()每运行一次自动会转向下一行读，在初始化时已经读了一次（第一行作为文件信息self.info），因此在跑next()时会读第二行。

|  |
| --- |
| **// SOURCE CODE**  // class FastaSequence:  def \_\_init\_\_(self, filename):  self.f = open(filename, 'r')  self.buf = ''  self.info = self.f.readline()  self.pos = 0  def \_\_iter\_\_(self):  return self  def next(self):  while '' == self.buf:  self.buf = self.f.readline()  print '' == self.buf  print self.buf  if '' == self.buf:  self.f.close()  raise StopIteration('StopIteration')  self.buf = self.buf.strip()  nextchar = self.buf[0]  self.buf = self.buf[1:]  self.pos += 1  return nextchar |
|  |
|  |

# powerful, worthwhile to use

try-except:

|  |
| --- |
| def findLargest(shapes):  """  Returns a tuple containing the elements of ShapeSet with the largest area.    shapes: ShapeSet    This is a sorting problem  """  result = (0,)  for each in shapes:  try:  if each.area() > result[-1].area():  result = (each,)  elif each.area() == result[-1].area():  result += (each,)  except AttributeError:  result = (each,)  return result |

|  |
| --- |
| def fastMaxVal(w,v,i,aW,memo):  global numCalls  numCalls=0  numCalls+=1  try: return memo[(i,aW)]  except KeyError:  if i==0:  if w[i]<=aW:  memo[(i,aW)]=v[i]  print(i,aW,v[i])  return v[i]  else:  memo[(i,aW)]=0  print(i,aW,0)  return 0  without\_i=fastMaxVal(w,v,i-1,aW,memo)  if w[i]>aW:  memo[(i,aW)]=without\_i  print(i,aW,without\_i)  return without\_i  else: with\_i=v[i]+fastMaxVal(w,v,i-1,aW-w[i],memo)  res=max(with\_i,without\_i)  print(i,aW,res)  memo[(i,aW)]=res  return res  def maxVal0(w,v,i,aW):  global memo  memo={}  return fastMaxVal(w,v,i,aW,memo) |

# 日积月累：

1，differences among \_\_str\_\_, \_\_repr\_\_,：

来源：<https://docs.python.org/2/reference/datamodel.html?highlight=__repr__#object.__repr__>

object.\_\_str\_\_(self)

Called by the str() built-in function and by the print statement to compute the “informal” string representation of an object. This differs from \_\_repr\_\_() in that it does not have to be a valid Python expression: a more convenient or concise representation may be used instead. The return value must be a string object.

|  |
| --- |
| 1. **>**>> class Student(object): 2. ... def \_\_init\_\_(self, name): 3. ... self.name = name 4. ... def \_\_str\_\_(self): 5. ... return 'Student object (name: %s)' % self.name 6. ... 7. >>> print(Student('Michael')) 8. Student object (name: Michael) |

若直接敲s的话：

|  |
| --- |
| 1. >>> s = Student('Michael') 2. >>> s |

object.\_\_repr\_\_(self)

Called by the repr() built-in function and by string conversions (reverse quotes) to compute the “official” string representation of an object. If at all possible, this should look like a valid Python expression that could be used to recreate an object with the same value (given an appropriate environment). If this is not possible, a string of the form <...some useful description...> should be returned. The return value must be a string object. If a class defines \_\_repr\_\_() but not \_\_str\_\_(), then \_\_repr\_\_() is also used when an “informal” string representation of instances of that class is required.

|  |
| --- |
| 1. **class Student(object):** 2. **def** **\_\_init\_\_**(self, name): 3. self.name = name 4. **def** **\_\_str\_\_**(self): 5. **return** 'Student object (name=%s)' % self.name 6. \_\_repr\_\_ = \_\_str\_\_ |

This is typically used for debugging, so it is important that the representation is information-rich and unambiguous.