"How many different ways can we find to...."

2017 CAS National Conference (Birmingham)

Session 4 – 1435hrs- 1515hrs

John Palmer

COMPUTING AT SCHOOL

EDUCATE · ENGAGE · ENCOURAGE

John Palmer

Regional Coordinator
CAS West Midlands CRC
CAS 3 Counties Hub Leader
CAS MT
CAS Board Member

Faculty Leader: Computing, Business & Vocational
The Chase, Malvern

About this “How many different ways can we find to....” session...

It’s not exactly Python Skills CPD, although hopefully you’ll learn something new

It’s for people who are teaching GCSE Computer Science and are happy with the basic coding ideas

Works best when people work in PAIRS!

It’s a practical session, so LOG ON and load Python / IDLE

I normally deliver it as a 2 hour session to Teachers, so this is the express version!

I deliver it to learners “longitudinally” over a month or so?

"In fifteen years we'll be teaching programming just like reading and writing . . . and wondering why we didn't do it sooner."

— Mark Zuckerberg

Try an Hour of Code for Computer Science Education Week December 9–15.
Anybody can learn!

http://code.org

Computer Science **Education Week**
DECEMBER 9–15, 2013

C O D E

Why is this “How many different ways can we find to....” idea important.....?

NEA / Controlled Assessment rules are changing.

Controlled Assessment is being “over taught” by some centres - leading to solutions that are causing problems at moderation because they are too similar.

Some teachers also seem to think some problems have just “one solution”, which has been seen to result in a lack of, even hostility to, diversity of solution!

New NEA rules are much more strict regarding teacher input, you are advised to develop activities that encourage a much more independent approach.

Summary of Dweck's *Mindset*

Fixed vs Growth

ability is static	ability is developed
avoids challenges	embraces challenges
gives up easily	persists in obstacles
sees effort as fruitless	sees effort as necessary
ignores useful criticism	learns from criticism
threatened by others	inspired by others' success

Summary...

Learners need to be taught language syntax

Learners need to develop algorithmic, problem solving skills...

Learners need opportunities to practice, be creative and value different approaches

Be wary of “model solutions” to NEA / Practice NEA tasks – there is no “right” way, although there may be “better” ways

We need to make learners better, more independent coders

“How many different ways can we find to....”

Lets start of by looking at the

LIST

Syntax Problems…….????

```
reindeer = [‘Dasher, ‘Dancer’ ‘Prancer’, ‘Rudolph’]

len[‘reindeer’]
Print(Reindeer)
for name in reindeer
        print(‘name’)
print(reindeer[5])
reindeer.append(Vixen)
reindeer.Insert(0, “Cupid”)
reindeer = ()
sort(reindeer)
print(sorted(reindeer)
```

Correct List Syntax…….

```
reindeer = [‘Dasher’, ‘Dancer’, ‘Prancer’, ‘Rudolph’]


len(reindeer)
print(reindeer)
for name in reindeer:
        print(name)
print(reindeer[2])
print(reindeer[0:2])
reindeer.append(“Vixen”)
reindeer.insert(0, “Cupid”)
reindeer = []
reindeer.sort()
print(sorted(reindeer))
```

Activity #1 – "List Printing"…….

```
reindeer = ['Dasher', 'Dancer', 'Prancer', 'Rudolph']
```

#How many different ways are there of printing out all the reindeer names in this list…. ????

Lists.......

```
reindeer = [‘Dasher’, ‘Dancer’, ‘Prancer’, ‘Rudolph’]

print(reindeer)

print(reindeer[0], reindeer[1], reindeer[2], reindeer[3])

for name in reindeer:
	print(name)

for i in range(0, len(reindeer)):
	print(reindeer[i])

i = 0
while i < len(reindeer):
	print(reindeer[i])
	i = i + 1
```

Activity #2

In pairs

Code as many different ways as you can for reversing the following list

L = ['dog', 'cat', 'man', 'bat']

There are a number of ways to do this, but it's very important to realise that these are ***not*** necessarily equivalent.

Activity #2

```
print("Original list")
L = ['dog', 'cat', 'man', 'bat']
print(L)

print ("#1")
print("Simple way to reverse the list - L.reverse()")
print(L)
L.reverse()
print(L)

#another simple way to do this...
L = ['dog', 'cat', 'man', 'bat']
print ("#2")
print("Simple way to reverse the list - list(reversed(L)")
print(list(reversed(L)))
```

Activity #2 Solutions

```
L = ['dog', 'cat', 'man', 'bat']

#1
Quick and dirty
N = [‘’,’’,’’,’’]  or N = [‘’]*len(L)
N[0] = L[3]
N[1] = L[2]
N[2] = L[1]
N[3] = L[0]

#2
#again!
print(L[3], L[2], L[1], L[0])
```

Activity #2 – Solutions

```
L = ['dog', 'cat', 'man', 'bat']


#3
N=[]
for i in range(0, len(L)):
        N.append(L[len(L) – 1 – i])
print(N)

#Loads of variations on this! Up / down etc..
```

Activity #2 – Solutions

```
L = ['dog', 'cat', 'man', 'bat']

#4
#pycon ninja style
N = L[::-1]
print(N)

#5
#while loop, working backwards
N = []
counter = len(L)-1
while counter >= 0:
		N.append(L[counter])
		counter = counter -1
print(N)
```

Activity #2 – Solutions

```
L = ['dog', 'cat', 'man', 'bat']

#6
#using for loop and working backwards
N = []
counter = len(L)-1
for index in range(counter, -1, -1):
        N.append(L[index])
print(N)

#7
#insert in position 0 on new list
N = []
for item in L:
   N.insert(0,item)
print(N)
```

Activity #2 – Solutions

```
L = ['dog', 'cat', 'man', 'bat']

#8
#insert in position 0 of new list
N = []
i = 0
while (i < len(L)):
    N.insert(0,L[i])
    i = i + 1
print(N)

#9
#Popping off a stack
N = []
while (len(L) > 0):
    N.append(L.pop())
print(N)
```

Activity #2 Summary

Thinking algorithmically....

#repeatedly insert items at index 0 of new list, N
#use a stack, "pop" things off onto a new list, N
#use different loop types – for / while etc...??
#use "slice" notation!
#Recursion
#Lambda operator

Getting learners to find as many different ways as they can is interesting, worthwhile, fun and definitely "Stretch and Challenge"!

Any questions?