Iteration II

Meta Activity: Team Disruptions

Common disruptions to learning in teams include: talking about topics that are off-task, teammates answering questions on their own, entire teams working alone, limited or no communication between teammates, arguing or being disrespectful, rushing to complete the activity, not being an active teammate, not coming to a consensus about an answer, writing incomplete answers or explanations, ignoring ideas from one or more teammates.

1. Pick four of the disruptions listed above. For each one, find something from the role cards that could help improve the team's success. Use a different role for each disruption.

a) Manager:

limited communication between teammates

b) Presenter:

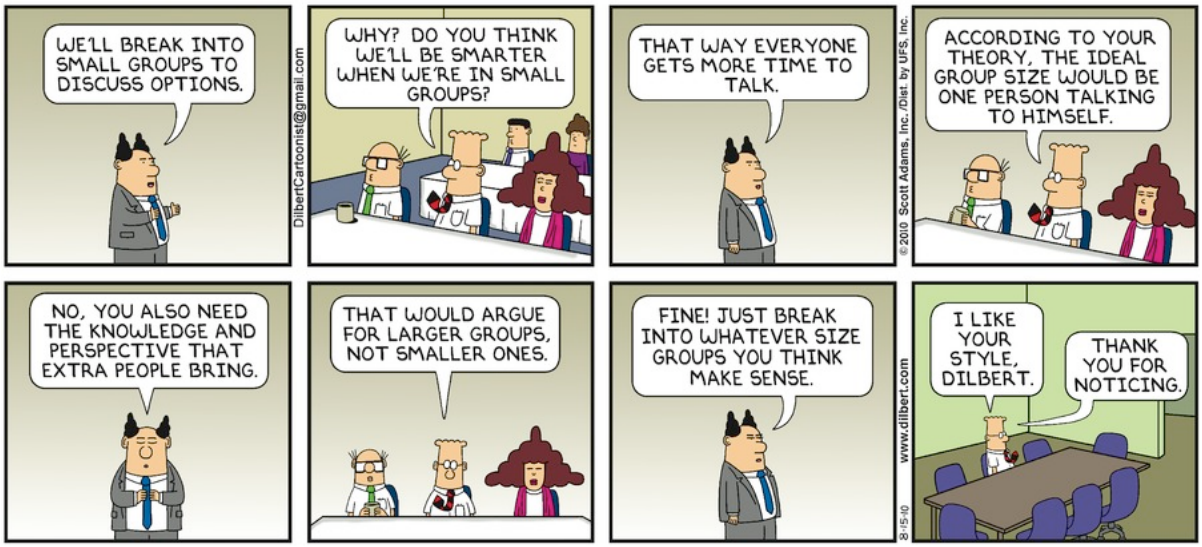
ignoring ideas from one or more teammates

c) Recorder:

writing incomplete answers or explanations

d) Reflector:

teammates answering questions on their own



Dilbert by Scott Adams. © Andrews McMeel Syndication. http://dilbert.com/strip/2010-08-15

Functions and `for` loops practice

2. Evaluate the following `for` loop:

```
def main() -> None:
    for i = 0 to 5:
        print(i, end = "")
main()
```

syntax error

3. What is the output of the following code snippet? Draw the loop table for each `for` loop.

```
def main() -> None:
    for i in range(4):
        print("i:", i+1)

    for j in range(4,0,-1):
        print("j: "+str(j))
main()
```

i	Output
0	i: 1
1	i: 2
2	i: 3
3	i: 4

j	Output
4	j: 4
3	j: 3
2	j: 2
1	j: 1

4. What is the output for `print_pattern('cat',3)`?

```
def print_pattern(word: str, copies: int) -> None:
    phrase = " * "
    for count in range(copies):
        phrase = phrase + word + " * "
        print(phrase)
```

count	phrase (after line 4)
0	`'* cat *'`
1	`'* cat * cat *'`
2	`'* cat * cat * cat *'`

5. What is the output for `print_pattern(3,'hi')`?

```
def print_pattern(cnt: int, instr: str) -> None:
    mystr = "Be"
    for i in range(cnt):
        mystr = mystr + instr * i + "!" * (cnt-i)
        print(mystr)
```

i	mystr (after line 4)
0	`'Be!!!'`
1	`'Be!!!hi!!'`
2	`'Be!!!hi!!hihi!'`

6. What is the output for `print_pattern(3)`?

```
def ants(repeat: int) -> None:
    ants = 1
    for i in range(repeat):
        print('The ants go marching', i, 'by', ants)
        ants = ants * 2
        print('hurrah ' * (i + 1))
    print('And ' + str(ants) + ' ants go marching down into the ground')
```

i	Output (line 4)	ants (after line 5)	Output (line 6)
0	`'The ants go marching 0 by 1'`	2	hurrah
1	`'The ants go marching 1 by 2'`	4	hurrah hurrah
2	`'The ants go marching 2 by 4'`	8	hurrah hurrah hurrah

```
The ants go marching 0 by 1
hurrah
The ants go marching 1 by 2
hurrah hurrah
The ants go marching 2 by 4
hurrah hurrah hurrah
And 8 ants go marching down into the ground
```

7. In honor of Valentine's day, write a ***lovely*** function called `valentine` that takes someone's name as a parameter, then prints it in a particular format. For example, for `'abi'`, the output must be:

```
I love a
I love b
I love i
I love a b i !
```

That is, the function should print, on separate lines, each letter of the name with "I love" before it. The function should also print one line at the very end with the full name of the person with a single space between each character of the name, followed by an exclamation point.

```
def valentine(name:str) -> None:
    name_w_spaces = ''
    for c in name:
        print("I love " + c)
        name_w_spaces += c + ' '
    print("I love " + name_w_spaces + "!")
```

8. Write a function called `cheer` that takes someone's name as a parameter and prints a cheer in a particular format. For example, for `'Fred'`, the output must be:

```
F !
rr !
eee !
dddd !
```

That is, the function should print a number of copies of each letter of the name on a separate line. The number of copies should increase with the number of lines printed. Each line should end with an exclamation point – and there should be no space between the exclamation point and the letters of the name. **Hint**: nested loops are not needed for this problem.

```
def cheer(name:str) -> None:
    count = 1
    for character in name:
        print(count * character + "!")
        count += 1
```

9. Write a function called `get_int` that takes in a list of digits, calculates and returns the number corresponding to the digits in the list. For example, `get_int([3,7,1])` must return `371`.

```
def get_int(digits: list) -> int:
    number=0
    for digit in digits:
      number = number * 10 + digit
    return number
```