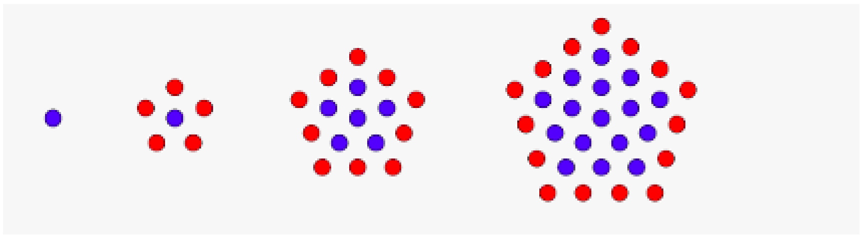
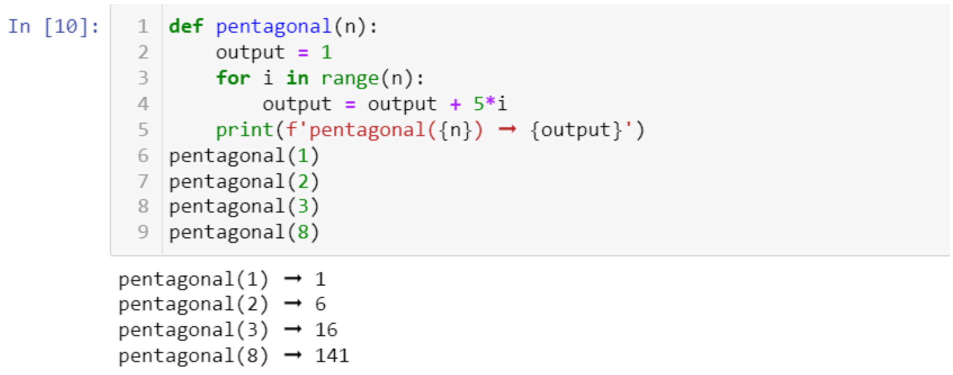
1. Write a function that takes a positive integer num and calculates how many dots exist in a pentagonal shape around the center dot on the Nth iteration.

In the image below you can see the first iteration is only a single dot. On the second, there are 6 dots. On the third, there are 16 dots, and on the fourth there are 31 dots.



Examples:
pentagonal(1) ➞ 1
pentagonal(2) ➞ 6
pentagonal(3) ➞ 16
pentagonal(8) ➞ 141

Ans:



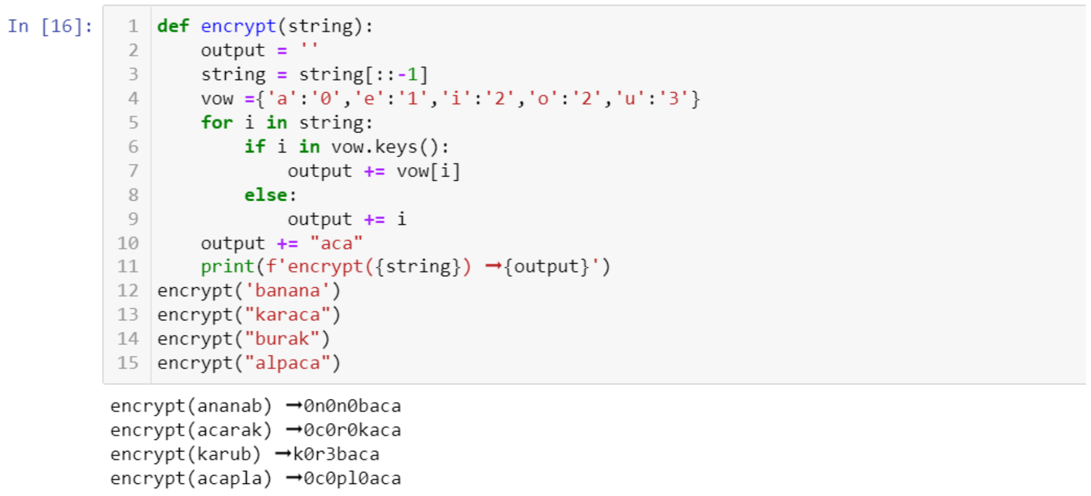
```
In [10]:
def pentagonal(n):
    output = 1
    for i in range(n):
        output = output + 5*i
    print(f'pentagonal({n}) ➞ {output}')
pentagonal(1)
pentagonal(2)
pentagonal(3)
pentagonal(8)
```

```
pentagonal(1) ➞ 1
pentagonal(2) ➞ 6
pentagonal(3) ➞ 16
pentagonal(8) ➞ 141
```

2. Make a function that encrypts a given input with these steps:

Input: "apple"
Step 1: Reverse the input: "elppa"
Step 2: Replace all vowels using the following chart:
a => 0
e => 1
i => 2
o => 2
u => 3
"1lpp0"
Step 3: Add "aca" to the end of the word: "1lpp0aca"
Output: "1lpp0aca"
Examples:
encrypt("banana") ➞ "0n0n0baca"
encrypt("karaca") ➞ "0c0r0kaca"
encrypt("burak") ➞ "k0r3baca"
encrypt("alpaca") ➞ "0c0pl0aca"

Ans:



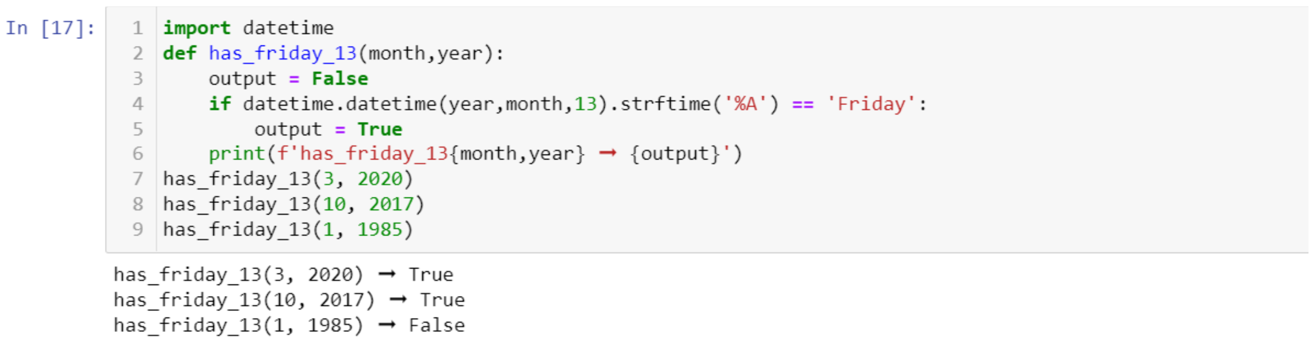
```
In [16]:
def encrypt(string):
    output = ''
    string = string[::-1]
    vow ={'a':'0','e':'1','i':'2','o':'2','u':'3'}
    for i in string:
        if i in vow.keys():
            output += vow[i]
        else:
            output += i
    output += "aca"
    print(f'encrypt({string}) ➞{output}')
encrypt('banana')
encrypt("karaca")
encrypt("burak")
encrypt("alpaca")
```

```
encrypt(ananab) ➞0n0n0baca
encrypt(acarak) ➞0c0r0kaca
encrypt(karub) ➞k0r3baca
encrypt(acapla) ➞0c0pl0aca
```

3. Given the month and year as numbers, return whether that month contains a Friday 13th.(i.e You can check Python's datetime module)

Examples:
has_friday_13(3, 2020) ➞ True
has_friday_13(10, 2017) ➞ True
has_friday_13(1, 1985) ➞ False

Ans:



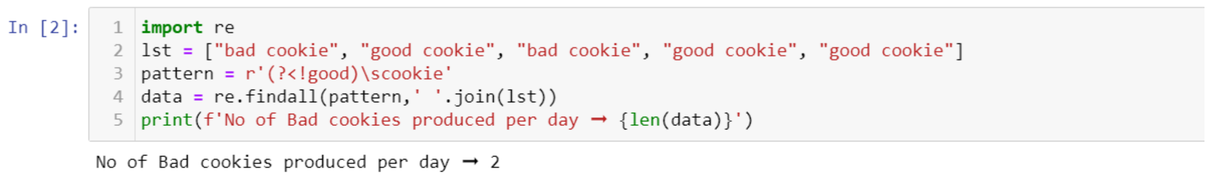
```
In [17]:
import datetime
def has_friday_13(month,year):
    output = False
    if datetime.datetime(year,month,13).strftime('%A') == 'Friday':
        output = True
    print(f'has_friday_13{month,year} ➞ {output}')
has_friday_13(3, 2020)
has_friday_13(10, 2017)
has_friday_13(1, 1985)
```

```
has_friday_13(3, 2020) ➞ True
has_friday_13(10, 2017) ➞ True
has_friday_13(1, 1985) ➞ False
```

4. Write a regular expression that will help us count how many bad cookies are produced every day. You must use RegEx negative lookbehind.

Examples:
lst = ["bad cookie", "good cookie", "bad cookie", "good cookie", "good cookie"]
pattern = "yourregularexpressionhere"
len(re.findall(pattern, ", ".join(lst))) ➞ 2

Ans:

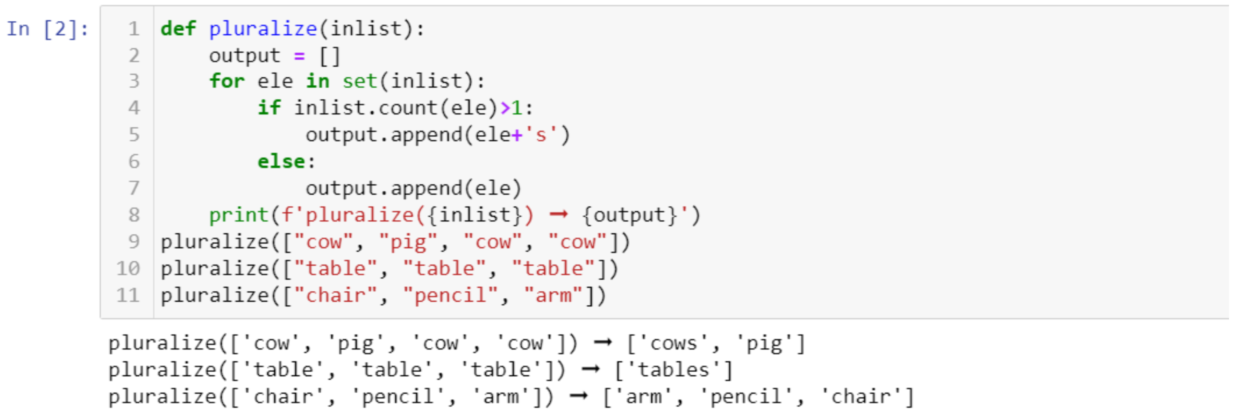


```
In [2]:
import re
lst = ["bad cookie", "good cookie", "bad cookie", "good cookie", "good cookie"]
pattern = r'(?<!good)\scookie'
data = re.findall(pattern,' '.join(lst))
print(f'No of Bad cookies produced per day ➞ {len(data)}')
```

```
No of Bad cookies produced per day ➞ 2
```

5. Given a list of words in the singular form, return a set of those words in the plural form if they appear more than once in the list.

Examples:
pluralize(["cow", "pig", "cow", "cow"]) ➞ { "cows", "pig" }
pluralize(["table", "table", "table"]) ➞ { "tables" }
pluralize(["chair", "pencil", "arm"]) ➞ { "chair", "pencil", "arm" }
Ans:



```
In [2]:
def pluralize(inlist):
    output = []
    for ele in set(inlist):
        if inlist.count(ele)>1:
            output.append(ele+'s')
        else:
            output.append(ele)
    print(f'pluralize({inlist}) ➞ {output}')
pluralize(["cow", "pig", "cow", "cow"])
pluralize(["table", "table", "table"])
pluralize(["chair", "pencil", "arm"])
```

```
pluralize(['cow', 'pig', 'cow', 'cow']) ➞ ['cows', 'pig']
pluralize(['table', 'table', 'table']) ➞ ['tables']
pluralize(['chair', 'pencil', 'arm']) ➞ ['arm', 'pencil', 'chair']
```