1. Enumerate sentence

Create a function that prints words within a sentence along with their index in front of the word itself.

For example if we give the function the argument "This is a sentence" it should print

```
1 This
2 is
3 a
4 sentence
```

In [ ]:

```
def enumWords(sentence):
    #Complete this method.
```

2. Fibonacci

Create a function `fibonacci()` which takes an integer `num` as an input and returns the first `num` fibonacci numbers.

Eg.

Input: `8`

Output: `[1, 1, 2, 3, 5, 8, 13, 21]`

Hint: You might want to recall [fibonacci numbers]

In [ ]:

```
def fibonacci(num):
    #Complete this method.

################ Checking code ######################
# Please don't edit this code
newList = fibonacci(10)
if newList == [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]:
    print("Success!")
else:
    print("Error! Your function returned")
    print(newList)
```

3. Guessing game 2

Ask the user to input a number and then have the program guess it. After each guess, the user must input whether it was too high, too low or the correct number. In the end, the program must always guess the users number and it must print out the number of guesses it needed.

In [ ]:

```

```

4. Find word

Create a function that searches for a word within a provided lists of words. Inputs to the function should be a list of words and a word to search for

The function should return `True` if the word is contained within the list and `False` otherwise.

In [ ]:

```
fruits = ["banana", "orange", "grapefruit", "lime", "lemon"]

def findWord(wordList, word):
    #Complete this method.

############### Checking code ######################
# Please don't edit this code

if findWord(fruits, "lime"):
    print("Success!")
else:
    print("Try again!")
```

5. Powers of 2

Use a while loop to find the largest power of 2 which is less than 30 million.

In [ ]:

```

```

6. Making a better school

This exercise is on defining classes. This topic is covered in the optional notebook python-intro-3-extra-classes.

Below is a copy of the `School`, `Student` and `Exam` classes, together with a copy of the code needed to populate an object of that class with students and exam results. Edit the `School` class to add in the following functions:

- **`.resits()` : this should return the list of exams that each student should resit if they get a "F" or "U" grade.**
- **`.prizeStudent()` : this should return the name of the student who scored the highest average percent across all of the exams.**
- **`.reviseCourse(threshold)` : this should return the name of the exam that gets the lowest average score across all students, if the average score is below `threshold`.**

Use these functions to find out which students need to resit which exams, which student should be awarded the annual school prize, and which courses should be revised as the average mark is less than 50%.

In [ ]:

```
class School:
    def __init__(self):
        self._students = {}
        self._exams = []

    def addStudent(self, name):
        self._students[name] = Student(name)

    def addExam(self, exam, max_score):
        self._exams.append(exam)

        for key in self._students.keys():
            self._students[key].addExam(exam, Exam(max_score))

    def addResult(self, name, exam, score):
        self._students[name].addResult(exam, score)

    def grades(self):
        grades = {}
        for name in self._students.keys():
            grades[name] = self._students[name].grades()
```

```
        return grades

# NOTE: This is not a class method
def addResults(school, exam, results):
    for student in results.keys():
        school.addResult(student, exam, results[student])


class Student:
    def __init__(self, name):
        self._exams = {}
        self._name = name

    def addExam(self, name, exam):
        self._exams[name] = exam

    def addResult(self, name, score):
        self._exams[name].setResult(score)

    def result(self, exam):
        return self._exams[exam].percent()

    def grade(self, exam):
        return self._exams[exam].grade()

    def grades(self):
        g = {}
        for exam in self._exams.keys():
            g[exam] = self.grade(exam)
        return g


class Exam:
    def __init__(self, max_score=100):
        self._max_score = max_score
        self._actual_score = 0

    def percent(self):
        return 100.0 * self._actual_score / self._max_score

    def setResult(self, score):
        if score < 0:
            self._actual_score = 0
        elif score > self._max_score:
            self._actual_score = self._max_score
        else:
            self._actual_score = score

    def grade(self):
        if self._actual_score == 0:
            return "U"
        elif self.percent() > 70.0:
            return "A"
        elif self.percent() > 60.0:
            return "B"
        elif self.percent() > 50.0:
            return "C"
        else:
            return "F"

# NOTE: This si not a class method
def addResults(school, exam, results):
    for student in results.keys():
        school.addResult(student, exam, results[student])
```

In [ ]:

```
school = School()

school.grades()
```

```
students = ["Andrew", "James", "Laura"]
exams = { "Maths" : 20, "Physics" : 50, "English": 30}
results = {"Maths" : {"Andrew" : 13, "James" : 17, "Laura" : 14},
           "Physics" : {"Andrew" : 34, "James" : 44, "Laura" : 27},
           "English" : {"Andrew" : 26, "James" : 14, "Laura" : 29}}

for student in students:
    school.addStudent(student)

for exam in exams.keys():
    school.addExam(exam, exams[exam])

for result in results.keys():
    addResults(school, result, results[result])

school.grades()
```