그런즉 너희가 먹든지 마시든지 무엇을 하든지 다 하나님의 영광을 위하여 하라 (고전10:31)

NOTE: The following materials have been compiled and adapted from the numerous sources including my own. Please help me to keep this tutorial up-to-date by reporting any issues or questions. Send any comments or criticisms to `idebtor@gmail.com` Your assistances and comments will be appreciated.

제 2-1 강: 함수와 뉴론$Function\ and\ Neuron$

학습 목표

- 함수와 뉴론을 이해한다.
- 인공뉴론과 인공신경망을 이해한다.
- JoyCoding: 첫 인공뉴론을 구현한다.

학습 내용

- 함수와 뉴론
- 인공뉴론과 인공신경망
- 인공뉴론의 구현

1. 함수

이 강의를 듣는 모든 수강생은 함수에 대해 공부한 적이 있습니다. 저는 중학교 때 처음으로 "함수"라는 단어를 처음 접했을텐데, '함수가 뭐지, 참 어렵다'고 생각했던 것만 기억이 납니다. 지금 생각하면 어려운 것도 아닌데 힘들었던 모양입니다. 아직도 조금은 기억이 나니까 말입니다.

그런데, 함수가 도대체 뭐죠?
수학적으로 정의하지 말고, 가볍게 생각하면, 함수에는 입력 x가 있고, 출력 y가 있습니다. 입력을 독립변수, 출력을 종속변수라 볼 수 있습니다. 입력 x의 값이 변함에 따라 출력 y의 값이 달라지는 관계입니다. 이 때, x와 y의 관계를 나타내는 식이 **함수** 입니다.

다음 그림은 함수 $y = f(x)$를 표현한 것입니다. 즉 함수 f는 입력 x를 받아 y를 출력합니다.

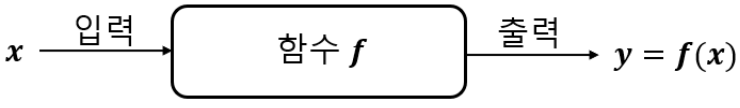


그림 2: 함수 $y = f(x)$의 모형

예를 들면, 섭씨 온도를 화씨로 변환하는 함수는 다음과 같이 정의할 수 있습니다.

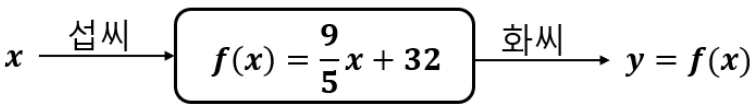


그림 3: 온도변환 함수 $y = f(x)$의 모형

과학자들은 다양한 자연 현상이나 우리가 필요한 것마다 수학적으로 모델링하여 **함수**를 정의해 왔습니다.

예를 들면, 현대통신의 시작과 끝이라고 말할 정도로 광범위하게 쓰이는 프리에^{Fourier} 변환도 다음과 같은 **함수**에 의해 이루어집니다. 푸리에 함수는 시간의 대한 입력 값을 받아 주파수 영역의 값으로 변환할 수 있는 함수입니다. 이 함수 덕분에 우리가 방송을 하고 휴대폰으로 전화를 할 수 있는 것입니다.

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{2\pi i \left(\frac{n}{T}\right) x} \tag{1}$$

우리가 영상이나 이미지를 휴대전화에서도 다룰 수 있는 것은 분량이 큰 이미지나 영상을 압축해서 작은 크기로 변환하고, 다시 복원할 수 있는 함수를 만들어낼 수 있어서 가능한 것입니다. 과학자들이 그런 작용을 할 수 있는 아래와 같은 코사인^{cosine} 변환이라는 함수를 찾아낸 것입니다.

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} f(i,j) cos\left[\frac{\pi(2i+1)u}{2N}\right] cos\left[\frac{\pi(2j+1)v}{2M}\right] \tag{2}$$

이렇게 복잡한 함수도 찾아내서 이미지나 영상을 압축해서 우리가 오늘 사용하고 있습니다.

그러나 최근 기계학습이나 딥러닝이 나오기까지 과학자들이 만들어내지 못했던 함수들 중에 하나는 많은 사진이나 동영상에서 고양이를 찾아내는 것입니다. 너무나도 다양한 경우의 수가 있기 때문에 이를 만족하는 **함수** 를 작성하기 매우 힘들었지요. 사진들을 입력으로 받아 사람이 울고 있는지 웃고 있는지 구별하는 함수조차 만들 수 없었습니다.

물론, 한 때는 스탠포드 연구팀의 기계학습에서는 이 사진을 보고 어린 아이가 야구방망이를 잡고 있다고 했지만, 이제는 사람이 직접 찾아내는 것만큼 컴퓨터가 고양이를 찾아낼 수 있고, 사람이 웃고 있는지 울고 있는지 구별해낼 수도 있습니다. 기계학습/딥러닝 덕분이죠.

컴퓨터가 이런 일을 할 수 있도록 하기 위한 전제 조건은 무엇인가요? Data, Big Data가 문제입니다.

자료가 많아야 한다는 것입니다. 그래서, 이제는 세계적인 IT회사들이 경쟁적으로 수십 기가바이트 용량의 저장공간을 우리들에게 무료로 제공하면서 데이터를 올리라고 권하는 것입니다.

결론적으로 제가 개인적으로 만들어 낸 "기계학습의 정의"는 **Universal Function Generator**, 우리 말로 하면 **만능 함수 제조기** 혹은 **꿈의 함수 제조기**$^{dream\ function\ generator}$ 라고 부르고 싶습니다.

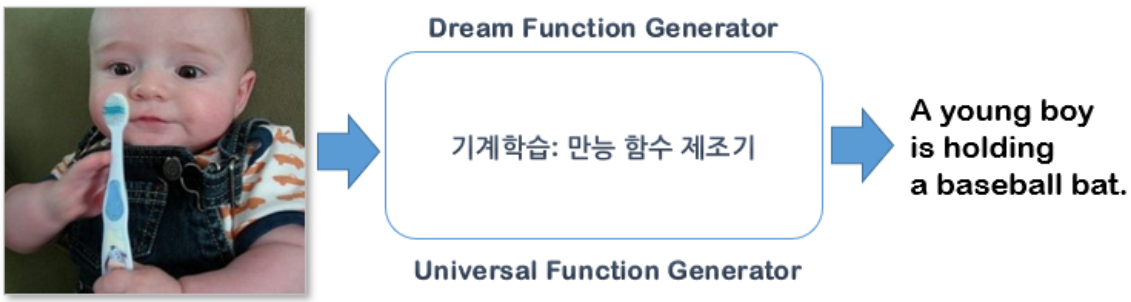


그림 4: 기계학습 - 만능 함수 제조기

2. 뉴론과 인공신경망

사람은 사물들을 빨리 인식하는데 컴퓨터는 그렇게 할 수 없다는 것을 과학자들은 이미 오래 전부터 알고 있었습니다. 그래서 과학자들은 사람의 뇌는 어떻게 그렇게 빨리 판단(계산)을 할 수 있는지 연구했습니다.

사람의 뇌는 약 850억개의 뉴론^{neuron} 즉 뇌세포로 구성되어 있다고 합니다. 2015년 호주에서 유레카 상을 수상한 이 멋진 사진을 보십시오.

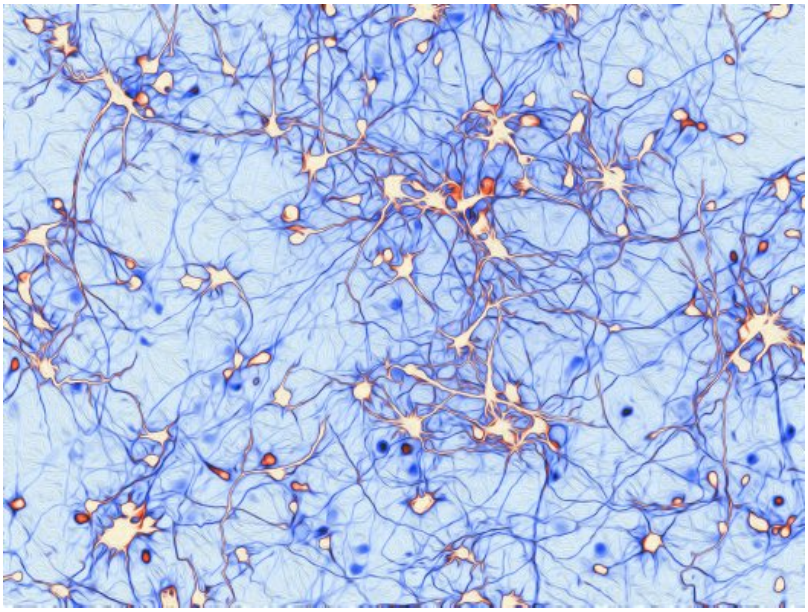


그림 5: 뉴론과 신경망

출처:[Australian Museum](https://australianmuseum.net.au/image/in-search-of-memory-eureka-prizes), 2015, Victor Anggono

뉴론들이 서로 연결된 망을 신경망$^{neural\ network}$이라고 합니다. 생물학적 뇌의 기본 단위인 뉴론들이 망^{network}으로 연결되어 서로 신호를 전달하면서 필요한 연산을 합니다. 뉴론을 서로 연결하는 시냅스의 수는 100조나 된다고 합니다. 그러니까 요즘 아무리 컴퓨터 기능이 뛰어나도, 병렬로 연결할지라도, 인간 한 사람의 두뇌에서 일어나는 신경망의 연산을 따라갈 수 없는 것입니다.

- In Search of Memory (Eureka Prizes)
- The neurocienteist on call

2.1 뉴론의 임계값

과학자들은 관찰을 통해 뉴론이 일종의 작은 하나의 연산자와 같다는 사실을 알아냈습니다. 뉴론은 입력 신호에 대해 즉시 반응하지는 않지만 출력을 내놓을 만큼 입력이 커질 때까지, 즉 임계값^{threshold}에 도달할 때까지 입력을 축적합니다. 예를 들면, 컵에 담긴 물이 처음으로 컵을 가득 채울 때까지는 흘러 넘치지 않는 것과 같습니다. 뉴론이 매우 작은 소리 신호가 아닌 아주 강한 신호 즉 어떤 임계값이 넘는 소리만을 전달한다고 이해하면 될 것입니다.

2.2 뉴론의 입력과 출력

과학자들이 발견한 또 다른 중요한 사실은 한 뉴런에 들어오는 신호 즉 입력은 다수이고 출력은 하나라는 것입니다. 둘째 사실은 여러 뉴론으로부터 전달되어 온 신호들은 합산되어 출력된다는 것이다. 합산된 값이 어떤 설정값 즉 임계값 이상이면 출력 신호가 있고, 이하이면 출력 신호가 없었습니다.

2.3 인공뉴론과 인공신경망

또한 인간의 생물학적 뉴론 하나가 아닌 다수가 연결되어 의미 있는 더 복잡한 작업을 할 수 있었습니다. 그래서, 과학자들은 뉴론과 신경망을 모델로 삼아, 단순한 계산을 하는 **인공뉴론** $artificial\ neuron$ 을 만들고, 여러 개의 인공뉴론들을 서로 연결한 **인공신경망**$artificial\ neural\ network$ 을 만들어서 계산해보니 예전에 할 수 없었던 계산들을 할 수 있었습니다.

다음은 두 개의 생물학적 뉴론과 하나의 인공 뉴론을 수학적 함수로 표현하여 모델링한 것입니다. 수상돌기는 입력 신호를 받고, 축색돌기는 신호를 출력합니다. 뉴론은 시냅스로 연결 되어 있어 시냅스에서 신호의 강약이 조정이 됩니다. 우리의 기억과 학습은 시냅스에 저장이 됩니다.

지금까지 우리가 이야기한대로 과학자들이 이러한 생물학적 뉴론과 신경망에서 영감을 받아, 여러 입력을 받아 연산을 한 후에 어떤 값을 출력하는 함수로 뉴론을 모델링을 하였습니다.

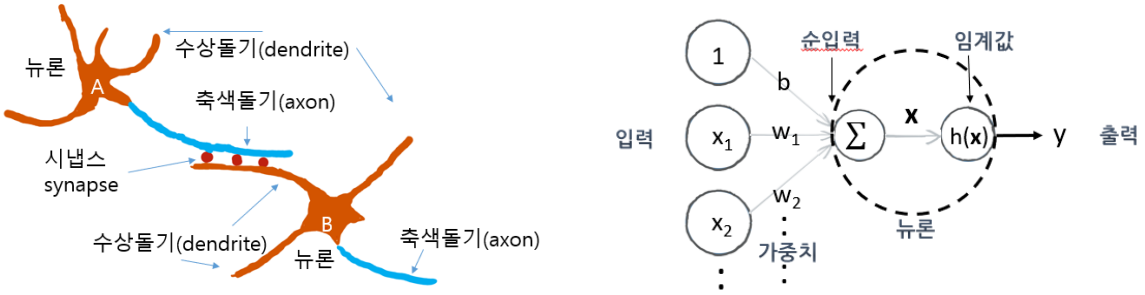


그림 6: 뉴론과 인공뉴론의 모델링

지금까지 반복해서 말씀드렸듯이 기계학습을 통해서 우리가 구하고자 하는 것은 입력과 출력의 관계를 함수로 표현하는 것입니다. 전통적인 프로그램은 결국 프로그래머가 함수의 정확한 역할을 정의하고 코딩을 하고, 컴퓨터는 이 함수를 실행하기만 하면 되는 것이었습니다. 컴퓨터는 프로그래머가 지정한 것 이상의 일을 할 수 없었지요.

다음 그림을 보면, 전통적인 프로그램이 아니라 우리는 인공신경망을 설계하고, 학습자료를 컴퓨터에게 넘겨 주면 컴퓨터가 기계학습을 하여 "꿈의 함수"를 만들어 냅니다. 프로그래머는 고양이 얼굴이 어떻게 생겼는지 귀가 큰지 작은지 특징을 코딩을 하는 것이 아니라 인공신경망을 설계하고, 학습자료를 제공하면 됩니다. 고양이를 구별해내고, 바둑 돌을 놓을 위치를 계산하는 함수는 컴퓨터가 찾아낼 것입니다.

그래서, 우리도 이제 인공신경망으로 함수들을 구현할 것입니다. 기존에 사용해왔던 수학적 표현이 아니라 뉴론들이 연결된 인공신경망을 학습시킬 것입니다. 다음 그림은 여러분이 궁극적으로 설계하고 구현할 인공신경망을 좀 간단히 그려 놓은 것입니다. 우리는 실제로 좀 더 복잡한 인공신경망을 만들어 손으로 쓴 숫자들을 구별하는 코딩을 해볼 것입니다.

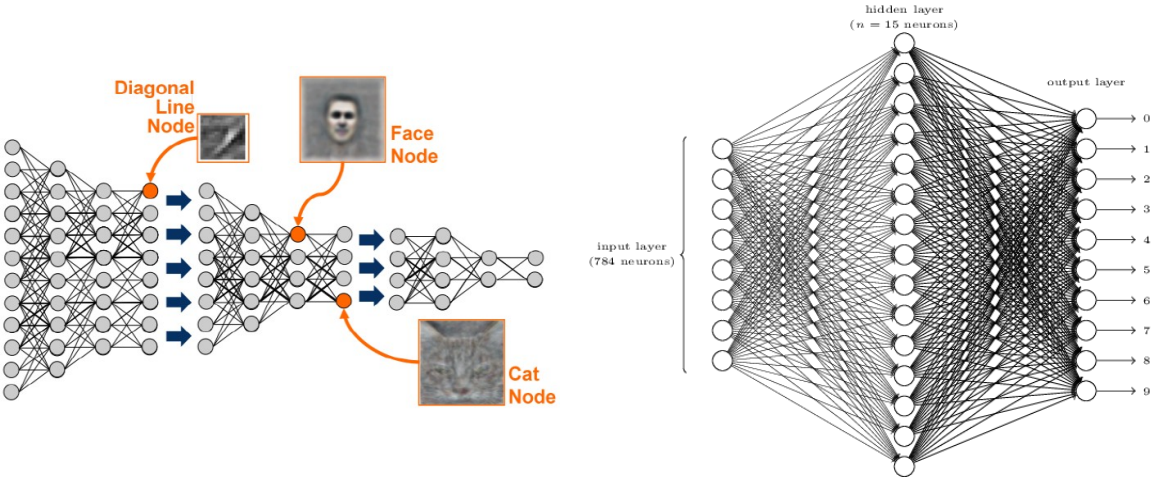


그림 7: 인공신경망(좌:이미지 분류, 우:MNIST 데이터셋)

출처(좌): Building high-level features using large scale unsupervised learning, 2012, Andrew Ng

3. 인공뉴론의 구현

천리길도 한 걸음부터라는 속담이 있죠?

인공신경망이 아니라 첫 인공뉴론을 만들어 간단한 인공뉴론 컴퓨팅부터 시작하고자 합니다. 아주 간단한 뉴론부터 시작하겠습니다.

인공뉴론을 만든다는 것은 무슨 말이죠? 함수로 같은 기능을 만들면 되는 것입니다. 마일($1\ mile$)을 킬로미터($1.61\ Km$)로 변환하는 함수를 뉴론으로 부르는 것입니다. 뉴론은 다음과 같이 원으로 표시하고 곱셈을 하는 뉴론이라고 가정합니다. f_x의 의미는 함수 f가 x를 매개변수(혹은 독립변수)로 가지고 있다는 뜻입니다. 간략히, 함수 f 는 x의 함수라고 말합니다.

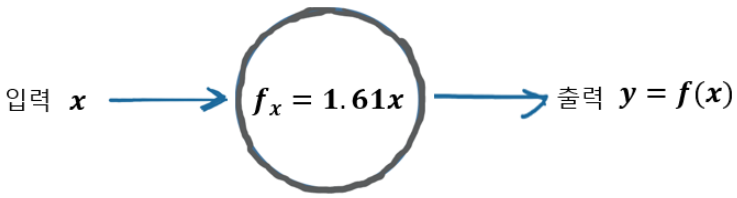


그림 8: 마일을 킬로미터로 변환하는 함수

"별로 인상적이 않네..."라고 여러분은 생각할지도 모릅니다. 그러나 괜찮습니다. 좀 더 흥미로운 개념의 기계학습으로 발전할 것이니까요. 잠시 후 여러분이 경험할 기계학습의 핵심인 신경망을 사용하기 전에, 지금은 간단한 예제를 사용한 것 뿐입니다.

파이썬에서 함수를 구현할 때, 함수 이름으로 f를 사용하는 것보다 함수의 기능을 알 수 있는 이름을 사용하는 것이 바람직 합니다. 그러면, 함수 이름을 `mileToKm` 로 정하여 함수를 구현하고, 함수를 몇 번 호출해보도록 하겠습니다. 한 가지 더 해볼 것은 함수를 여러 번 호출한 결과들을 시각화하는 코딩을 해보겠습니다.

3.1 거리 단위 변환 함수와 시각화 함수

파이썬 함수로 인공뉴론을 구현합니다. `mileToKm()` 라는 뉴론을 구현하고, 함수 `plotMileToKm()` 은 인공뉴론의 결과를 시각화하는데 도움이 되는 함수입니다. 시각화를 위해

서 `matplotlib.pyplot` 라이브러리를 import하며, 또한 결과를 쥬피터 노트북 셀 안에 그리기 위하여 `%matplotlib inline` 명령을 실행합니다.

- mileToKm(x) -- 입력 x 마일을 킬로미터로 변환한 값을 반환함. x의 값은 scalar 혹은 numpy 배열이 가능하나, list형식은 가능하지 않음.
- plotMileToKm(x, y) -- x, y의 값들에 대한 그래프를 출력함. x, y의 값은 list 형식 혹은 numpy 배열이 가능함. `fontsize` 는 옵션입니다.

In [1]:
```
import matplotlib.pyplot as plt
%matplotlib inline

def mileToKm(x):
    """ x 마일을 Km로 변환하여 반환"""
    return 1.61 * x

def plotMileToKm(x, y):
    """ x, y의 값들을 그래프로 출력 """
    plt.figure()
    plt.plot(x, y)
    plt.title('Mile to Km')
    plt.xlabel('Mile')
    plt.ylabel('Km')
    plt.show()
```

3.2 함수 호출

- 입력 변수를 정하고, 초기화 합니다. `myMile=2, urMile=3`
- 입력 변수로 `mileToKm()` 함수를 호출하고, 반환값을 `myKm, urKm` 에 각각 저장합니다.
- 반환받은 값들을 출력합니다.

In [2]:
```
myMile = 2
myKm = mileToKm(myMile)
print(myMile, 'Miles are', myKm, 'Km')
```

```
2 Miles are 3.22 Km
```

In [3]:
```
urMile = 3
urKm = mileToKm(urMile)
print(urMile, 'Miles are', urKm, 'Km')
```

```
3 Miles are 4.83 Km
```

3.3 for 루프를 사용한 함수 호출

함수를 여러 번 호출하기 위하여 `for` 루프를 사용합니다. 여러 `x` 값들, 즉 `x = 0, 1, 2, 3, 4` 의 경우들을 차례로 입력하고, 결과를 출력하는 코딩입니다.

새로운 방식으로 `print()` 를 사용해보았습니다. `print` 구문에서 `{}.format` 형식을 사용하면, 코딩이 좀 길어지긴 하지만, 읽기 쉬운 출력문을 작성할 수 있는 장점이 있습니다. 여러 항목을 출력할 때 사용하면 유익합니다.

In [4]:
```
for mile in range(0, 5):
    print('{}mi:{}km'.
          format(mile, mileToKm(mile)))
```

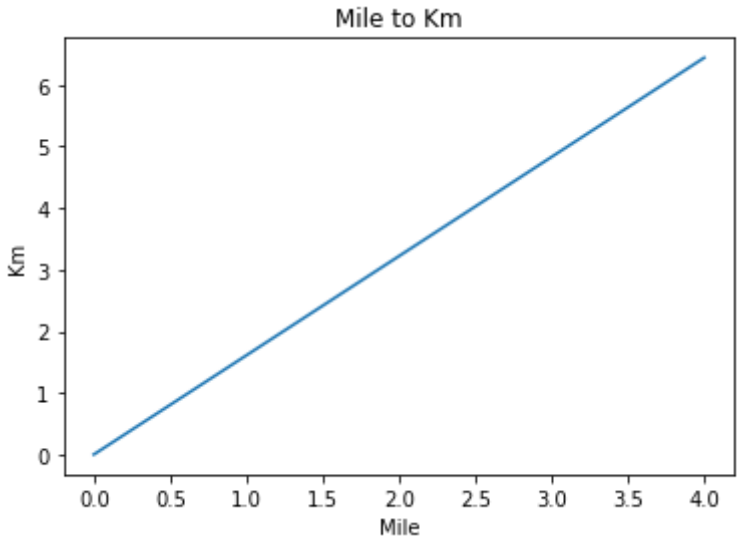
```
0mi:0.0km
1mi:1.61km
2mi:3.22km
3mi:4.83km
4mi:6.44km
```

3.4 함수 결과의 시각화

다른 분야와 마찬가지로 기계학습에서도 함수나 함수의 결과를 시각화하는 것은 중요한 부분입니다. 다음은 입력 `x = 0, 1, 2, 3, 4` 마일에 대한 킬로미터 값을 시각화한 것입니다.

`x, y` 의 값이 하나(scalar)가 아니고 괄호 `[ ]` 안에 나열되어 있습니다. 이를 `list` 형식^{type}이라고 부릅니다. 이러한 `list` 형식을 `plot(x, y)` 함수가 받아서 다음과 같이 시각화를 할 수 있습니다.

In [5]:
```
x = [0, 1, 2, 3, 4]
y = [0, 1.61, 3.22, 4.83, 6.44]
plotMileToKm(x, y)
```



3.4.1 데이터 포인트 시각화

직선 대신 우리가 명시한 점(데이터 포인트)들을 표시하려면, `plot()` 함수에 인자를 더하면 됩니다.

함수 `plotMileToKm()` 안에서 `plot()` 함수의 세 번째 인자에,

- `ob` 를 추가하고 그 셀을 실행한 후, 다음 셀을 실행하면, 파란 동그란 점들이 출력이 되고,
  `plt.plot(x, y, 'ob')`
- `-sr` 을 추가하고 그 셀을 실행한 후, 다음 셀을 실행하면, 빨간 네모난 점들과 선이 연결되어 출력이 됩니다.
  `plt.plot(x, y, '-sr')`

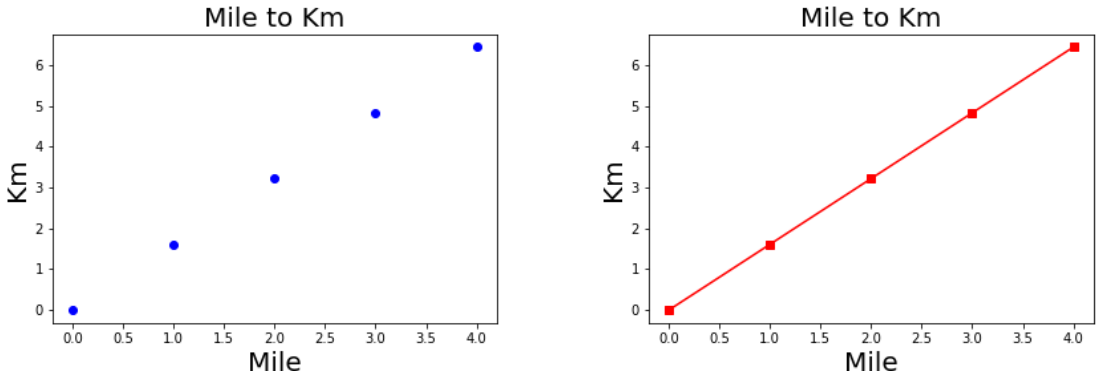
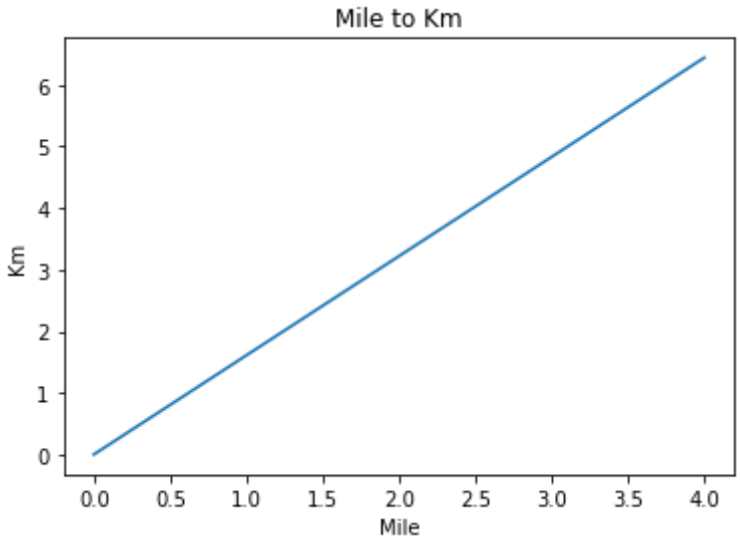


그림 9: mileToKm 함수 시각화 결과(ob, -sr)

ob 는 동그란 모양(o)의 파란색(blue) 표시^{marker}를 의미합니다. 여러 모양 (v, ^, <, >, ., s, p, x) 과 색 (b, c, m, r, y, k) 을 나타낼 수 있으니, 여기를 참조(모양), 참조(색)하십시오.

In [6]:
```
plotMileToKm(x, y)
```



3.5 list 형식^{type} 데이터 만들기

앞에서 우리는 x, y 값들을 하나씩 계산하고, 또한 list 형식으로 만들어서 시각화를 했습니다. x = [0, 1, 2, 3, 4] y = [0, 1.61, 3.22, 4.83, 6.44] 이러한 list 를 좀 더 쉽게 만들 수 있는 방법이 있습니다.

x 와 같이 일정한 패턴으로 진행되는 일련의 수를 list 로 만들기 위해서는 아래 두 함수 중에 하나를 이용합니다.

- range(start, end, step) -- start에서 end 까지 step 간격으로 숫자를 발생함(end는 포함하지 않음). step을 생략하면, 기본값은 step = 1.
- linspace(start, end, npoints) -- start에서 end 까지 npoints의 숫자를 발생함(end는 포함하지 않음)

3.5.1 for 루프 사용하기

y 값을 list 형식으로 구하려면 for 루프 혹은 list comprehension 이라는 파이썬 기능을 활용하면 좋습니다.

- `for` 루프에서 `y` 를 사용하기 전에 초기화해야 합니다.
- append()는 y와 같은 list 형식의 자료에 사용할 수 있는 list클래스의 메소드(함수)입니다.

In [7]:
```
y = []
for mile in range(0, 5):
    y.append(mileToKm(mile))
print(y)
```

```
[0.0, 1.61, 3.22, 4.83, 6.44]
```

3.5.2 리스트 컴프리헨션$^{list\ comprehension}$ 사용하기

리스트 컴프리헨션은 리스트를 만드는 짧고 간결한 방법입니다

다음 구문은 위의 for 루프 구문을 **리스트 컴프리헨션** 으로 변환한 것입니다.

```
y = [ mileToKm(mile) for mile in range(0, 5) ]
```

`for` 루프 부분은 같습니다. 다만, `for` 루프처럼 실행하면서, `mile` 변수로 `mileToKm(mile)` 을 호출합니다. 그리고 그 결과는 자동으로 list 안에 차례대로 저장 됩니다. 한 줄로 `for` 루프를 처리하는 것이 신기하죠? 많이 쓸모가 있을 것 같습니다.

In [8]:
```
y = [ mileToKm(mile) for mile in range(0, 5) ]
print(y)
```

```
[0.0, 1.61, 3.22, 4.83, 6.44]
```

3.6 NumPy^{넘파이}의 함수 호출

이제 넘파이를 활용해 보도록 하겠습니다. 넘파이를 사용하려면, `numpy` 모듈을 `import` 해야 합니다. `numpy` 를 줄여서 `np` 로 코딩하기 위하여 `import` 할 때, `as np` 를 붙여서 `import` 합니다. 아니면, `numpy` 함수를 호출할 때마다 `numpy.` 를 사용해야 합니다.

```
import numpy as np
```

리스트 형식의 x값을 numpy로 변환할 수 있고, 아니면 새로 만들 수도 있습니다.

```
x = [0, 1, 2, 3, 4, 5]
```

아래에서 `x` 는 `list` 형식이고, `xn` 은 넘파이 배열 형식입니다. 두 형식을 각각 출력하면 모양이 약간 다른 것에 유의하길 바랍니다.

In [9]:
```
import numpy as np

x = [0, 1, 2, 3, 4]       # list type
print(x)

xn = np.array(x)          # numpy array type
print(xn)

xn = np.array(np.arange(0, 5))   # newly generated xn
print(xn)
```

```
[0, 1, 2, 3, 4]
[0 1 2 3 4]
[0 1 2 3 4]
```

3.7 numpy 배열의 이점

자, `x` 값이 넘파이 배열로 준비되었다면, `y` 값을 어떻게 구할 수 있나요?

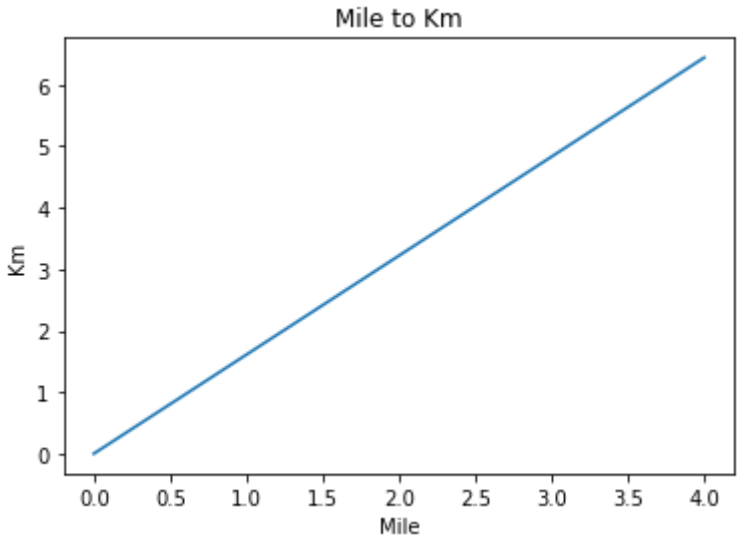
`for` 루프 혹은 리스트 컴프리헨션으로 구할 수 있습니다. 그러나, 아주 간편한 방법이 있습니다. 넘파일 배열 x를 그대로 `mileToKm(x)` 로 호출하는 것입니다.

그러면, 그 함수는 넘파이 배열에 나열된 순서대로 하나씩 계산해서 넘파이 배열 형식으로 반환합니다. 와우~~ 놀랍죠?

코딩해서 테스트 해볼까요?

In [10]:
```
x = np.arange(0, 5)                 # x -- numpy array
y = mileToKm(x)                     # y -- numpy array
print(x)
print(y)
plotMileToKm(x, y)                  # pass x, y numpy array
```

```
[0 1 2 3 4]
[0.   1.61 3.22 4.83 6.44]
```



코딩이 좀 복잡한가요? 그래도, 이번 코딩을 꼭 연습해 보길 바랍니다.

코딩은 강의를 듣거나 보거나, 묵상하거나 토론해서 배울 수 없습니다. 코딩을 배우는 방법은 손코딩이 최고입니다. 코딩을 하지 않고 기계학습을 배우는 방법은 없습니다.

In [11]:
```
f = [9/5 * c + 32 for c in range(-10, 101, 10)]
f
```

Out[11]:
```
[14.0, 32.0, 50.0, 68.0, 86.0, 104.0, 122.0, 140.0, 158.0, 176.0, 194.0, 212.0]
```

참고자료

- CS231n Convolutional Neural Networks for Visual Recognition, Python Numpy Tutorial, Stanford University
- Python For Data Science Cheat Sheet NumPy Basics, DataCamp
- Python Numpy Tutorial - http://cs231n.github.io/
- 김태완 블로그: 파이썬 데이터 사이언스 Cheat Sheet

학습 정리

- 함수와 뉴론의 이해
- 인공뉴론과 인공신경망의 이해
- 첫 인공뉴론의 구현

In [ ]: