In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

DATA

In [18]:

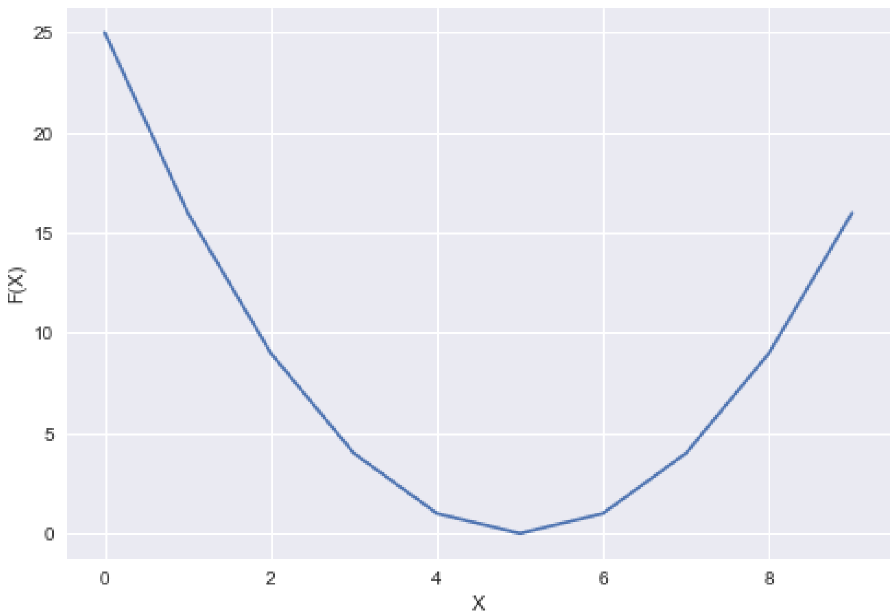
```
X = np.arange(10)
Y = (X - 5) ** 2

print(X, Y)
print(X.shape, Y.shape)
```

```
[0 1 2 3 4 5 6 7 8 9] [25 16  9  4  1  0  1  4  9 16]
(10,) (10,)
```

Visualise the data

In [16]:

```
plt.style.use('seaborn')
plt.plot(X, Y)
plt.xlabel('X')
plt.ylabel('F(X)')
plt.show()
```



Gradient Descent

Goal Given a function F(X), we want to find the value of x that minimizes F(X)

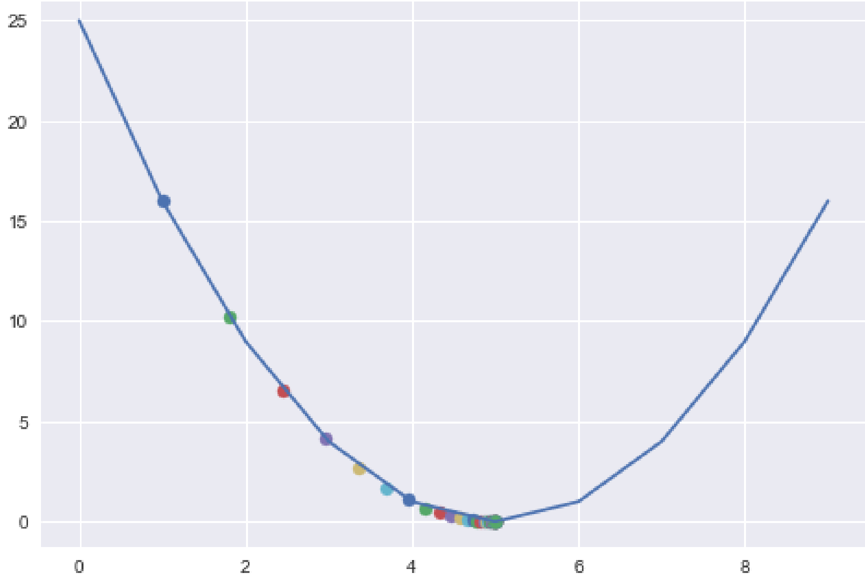
In [7]:

```
x = 0
lr = 0.1

# Take 50 steps in downhill direction
for i in range(50):
    grad = 2 * (x - 5)
    x = x - lr * grad
    print(x)
```

In [19]:

```
x = 0
lr = 0.1

errors = []

plt.plot(X, Y)

# Take 50 steps in downhill direction
for i in range(50):
    grad = 2 * (x - 5)
    x = x - lr * grad
    y = (x - 5) ** 2
    errors.append(y)
    print(x)
    plt.scatter(x, y)

plt.show()
```

```
1.0
1.8
2.4400000000000004
2.9520000000000004
3.3616
3.68928
3.9514240000000003
4.1611392
4.32891136
4.4631290880000005
4.570503270400001
4.65640261632
4.725122093056
4.7800976744448
4.82407813955584
4.859262511644672
4.8874100093157375
4.90992800745259
4.927942405962073
4.942353924769658
4.953883139815726
4.9631065118525814
4.9704852094820655
4.976388167585652
4.981110534068522
4.984888427254818
4.987910741803854
4.990328593443083
4.992262874754466
4.993810299803573
4.995048239842858
4.996038591874287
4.996830873499429
4.9974646987995435
4.997971759039634
4.9983774072317075
4.998701925785366
4.998961540628293
4.999169232502634
4.999335386002107
4.999468308801686
4.9995746470413485
4.999659717633079
4.9997277741064625
4.99978221928517
4.999825775428136
4.99986062034250 9
4.999888496274007
4.999910797019206
4.999928637615365
```

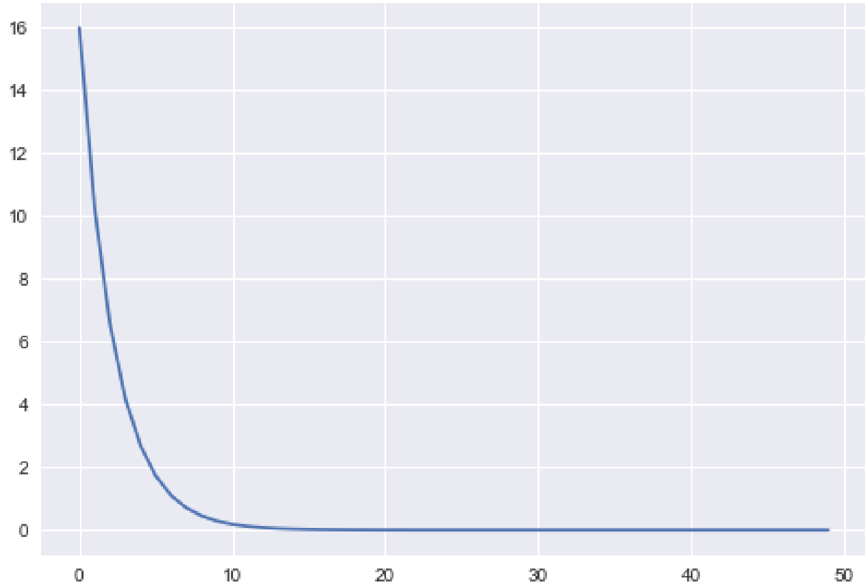


In [20]:

```
plt.plot(errors)
```

Out[20]:

```
[<matplotlib.lines.Line2D at 0x129bfce8>]
```



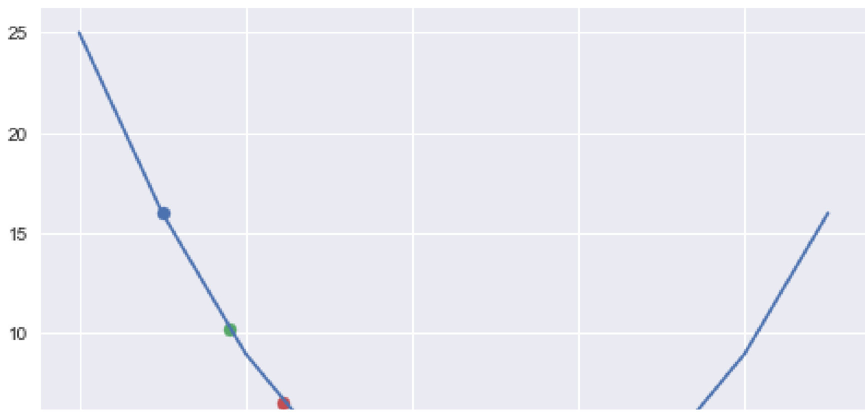
Experiment

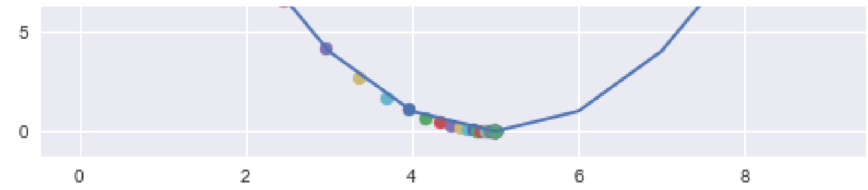
In [22]:

```
X = np.arange(10)
Y = (X - 5) ** 2

print(X, Y)
print(X.shape, Y.shape)

x = 0
lr = 0.1

errors = []

plt.plot(X, Y)

# Take 50 steps in downhill direction
for i in range(50):
    grad = 2 * (x - 5)
    x = x - lr * grad
    y = (x - 5) ** 2
```

```
    errors.append(y)
    print(x)
    plt.scatter(x, y)

plt.show()
```

```
[0 1 2 3 4 5 6 7 8 9] [25 16  9  4  1  0  1  4  9 16]
(10,) (10,)
1.0
1.8
2.4400000000000004
2.9520000000000004
3.3616
3.68928
3.9514240000000003
4.1611392
4.32891136
4.4631290880000005
4.570503270400001
4.65640261632
4.725122093056
4.7800976744448
4.82407813955584
4.859262511644672
4.8874100093157375
4.90992800745259
4.927942405962073
4.942353924769658
4.953883139815726
4.9631065118525814
4.9704852094820655
4.976388167585652
4.981110534068522
4.984888427254818
4.987910741803854
4.990328593443083
4.992262874754466
4.993810299803573
4.995048239842858
4.996038591874287
4.996830873499429
4.9974646987995435
4.997971759039634
4.9983774072317075
4.998701925785366
4.998961540628293
4.999169232502634
4.999335386002107
4.999468308801686
4.9995746470413485
4.999659717633079
4.9997277741064625
4.99978221928517
4.999825775428136
4.999860620342509
4.999888496274007
4.999910797019206
4.999928637615365
```

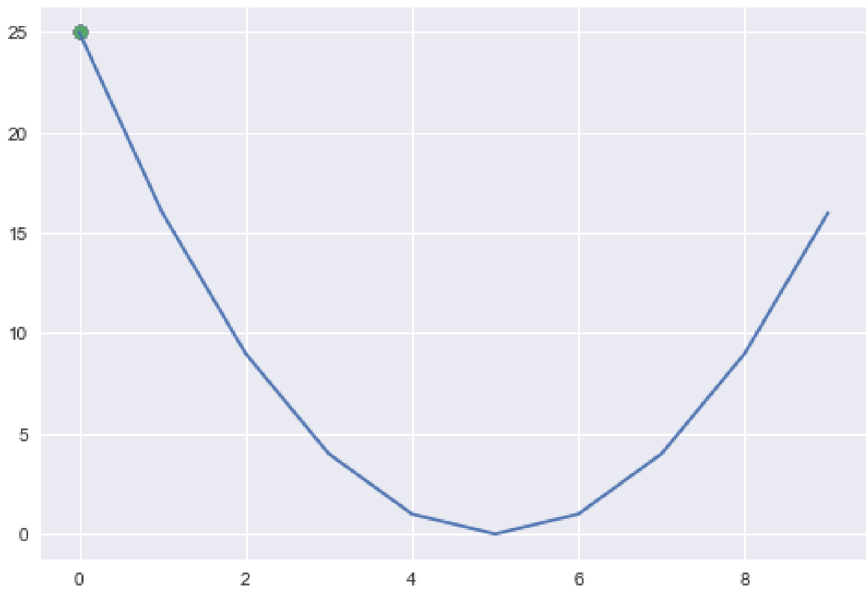




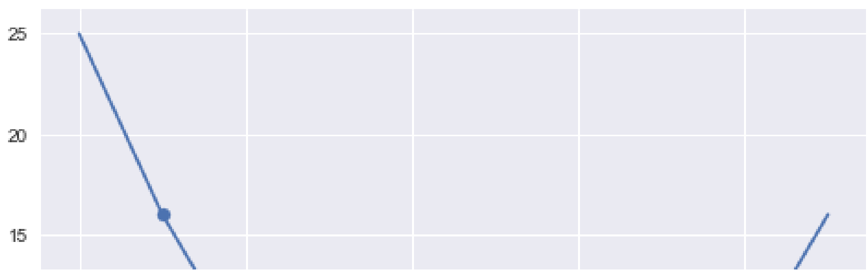
In [25]:

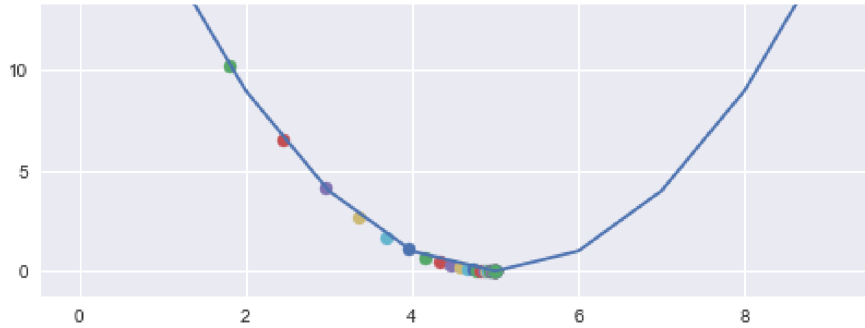
```
i = 0

for j in range(0, 11, 1):

    X = np.arange(10)
    Y = (X - 5) ** 2

    print(X, Y)
    print(X.shape, Y.shape)

    x = 0
    lr = j / 10
    errors = []
    plt.figure(i)
    plt.plot(X, Y)

    # Take 50 steps in downhill direction
    for i in range(50):
        grad = 2 * (x - 5)
        x = x - lr * grad
        y = (x - 5) ** 2
        errors.append(y)
#           print(x)
        plt.scatter(x, y)

    plt.show()
    i += 1
```

```
[0 1 2 3 4 5 6 7 8 9] [25 16  9  4  1  0  1  4  9 16]
(10,) (10,)
```

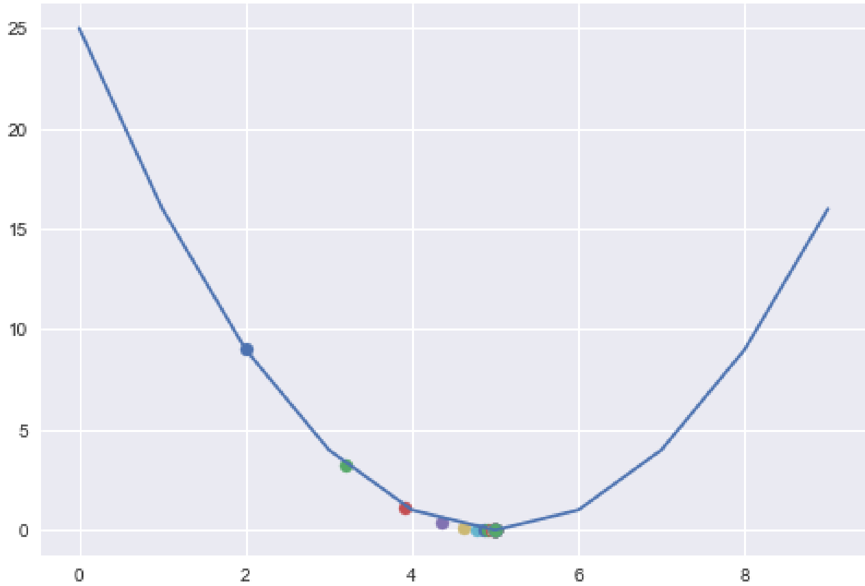



```
[0 1 2 3 4 5 6 7 8 9] [25 16  9  4  1  0  1  4  9 16]
(10,) (10,)
```

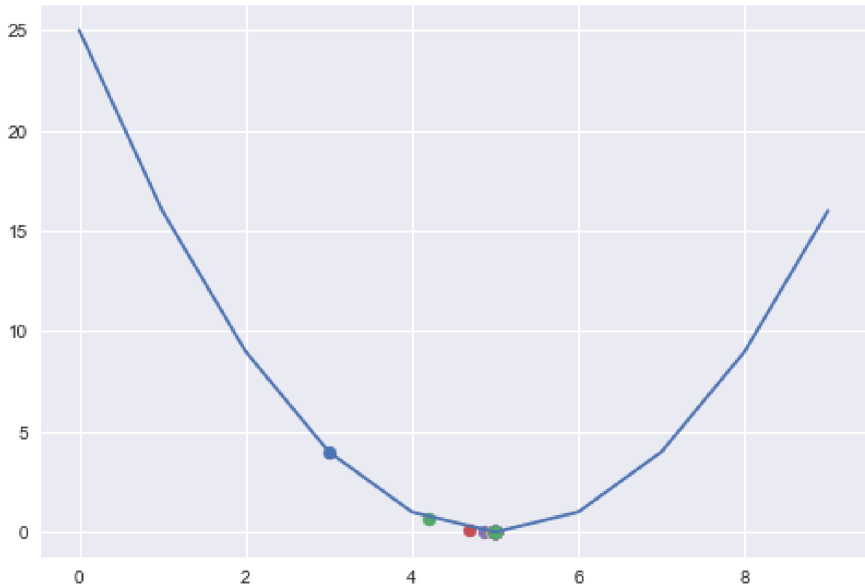




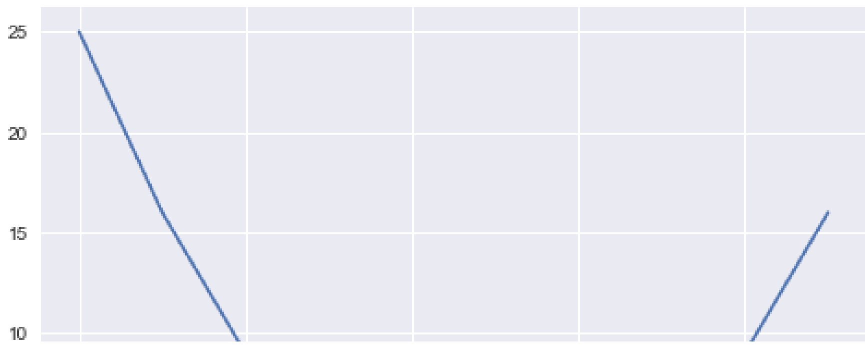

```
[0 1 2 3 4 5 6 7 8 9] [25 16  9  4  1  0  1  4  9 16]
(10,) (10,)
```

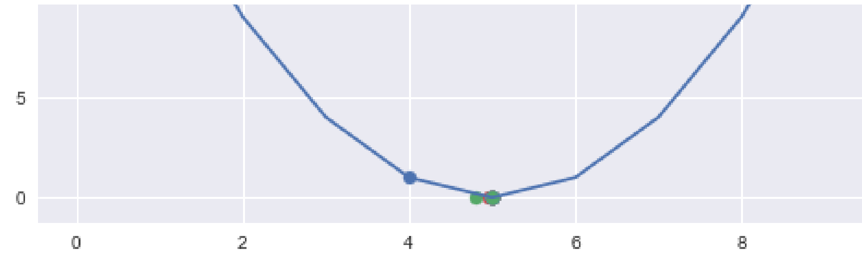



```
[0 1 2 3 4 5 6 7 8 9] [25 16  9  4  1  0  1  4  9 16]
(10,) (10,)
```

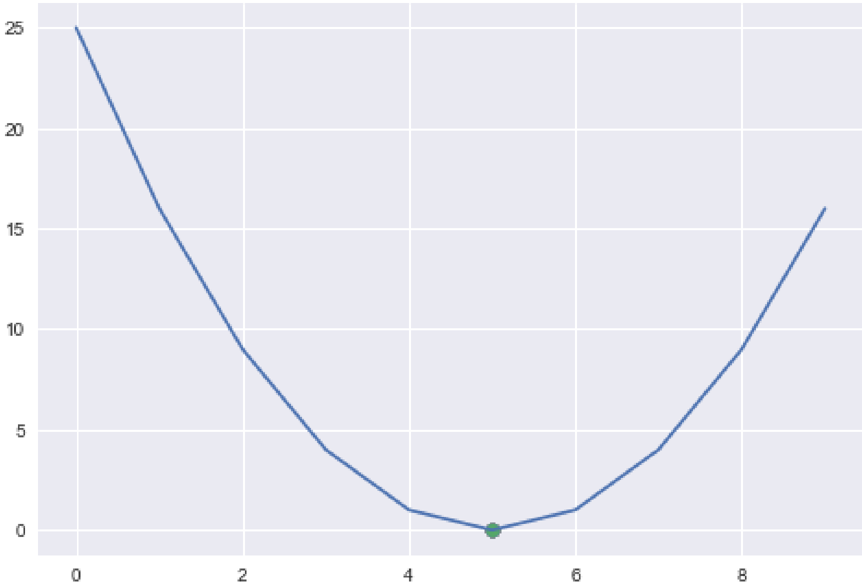



```
[0 1 2 3 4 5 6 7 8 9] [25 16  9  4  1  0  1  4  9 16]
(10,) (10,)
```

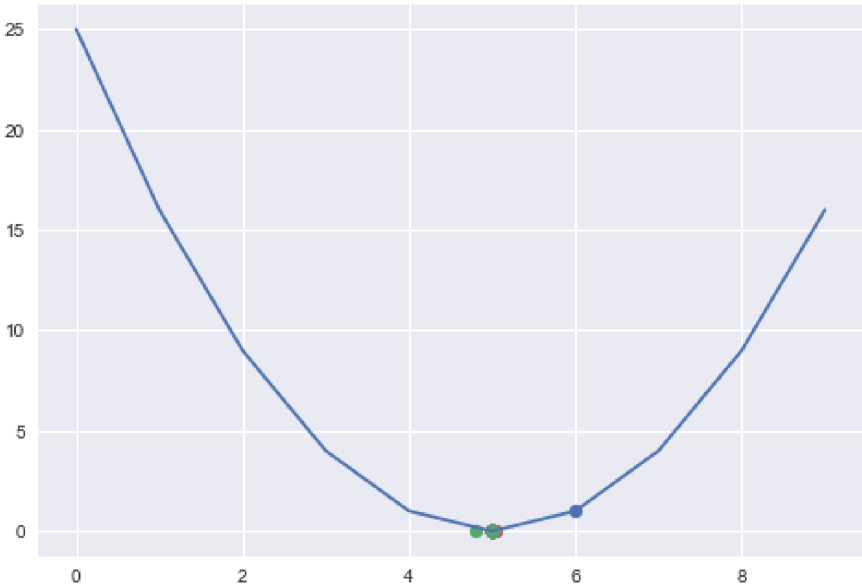




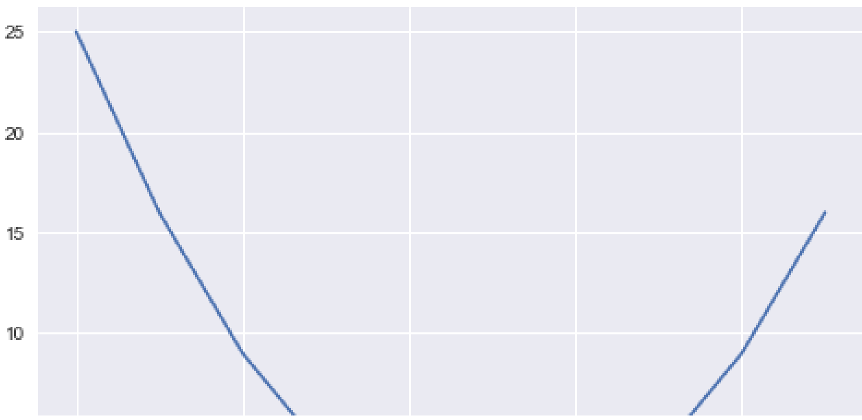

```
[0 1 2 3 4 5 6 7 8 9] [25 16  9  4  1  0  1  4  9 16]
(10,) (10,)
```

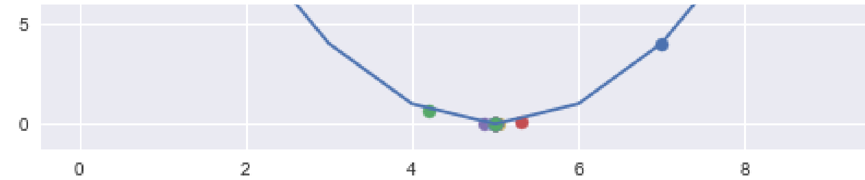



```
[0 1 2 3 4 5 6 7 8 9] [25 16  9  4  1  0  1  4  9 16]
(10,) (10,)
```

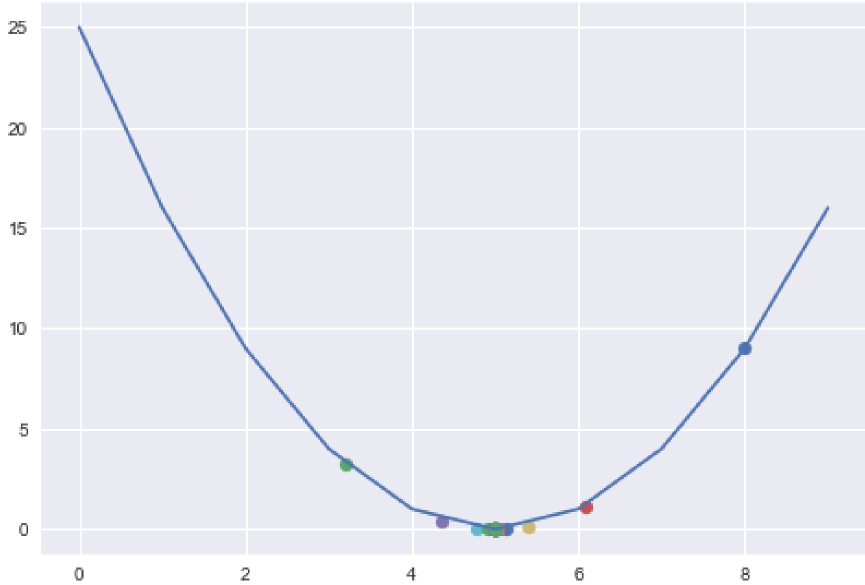



```
[0 1 2 3 4 5 6 7 8 9] [25 16  9  4  1  0  1  4  9 16]
(10,) (10,)
```

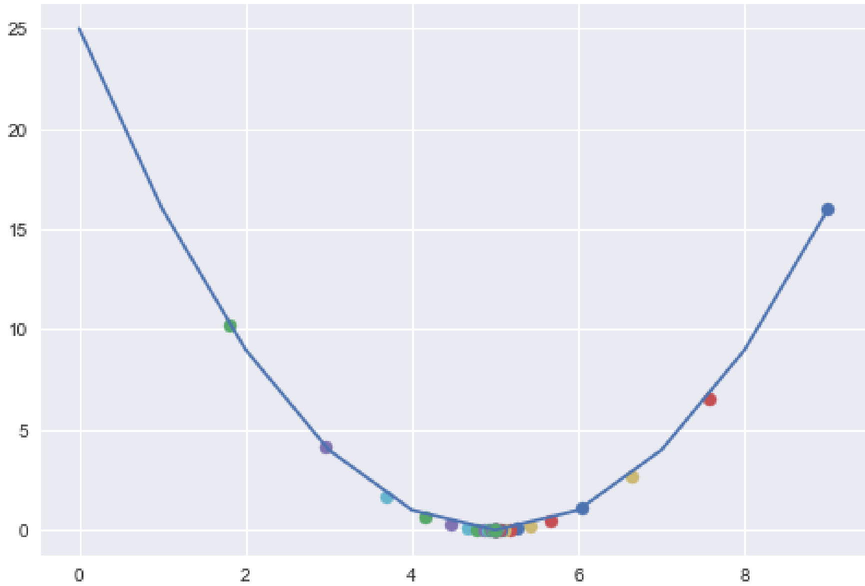




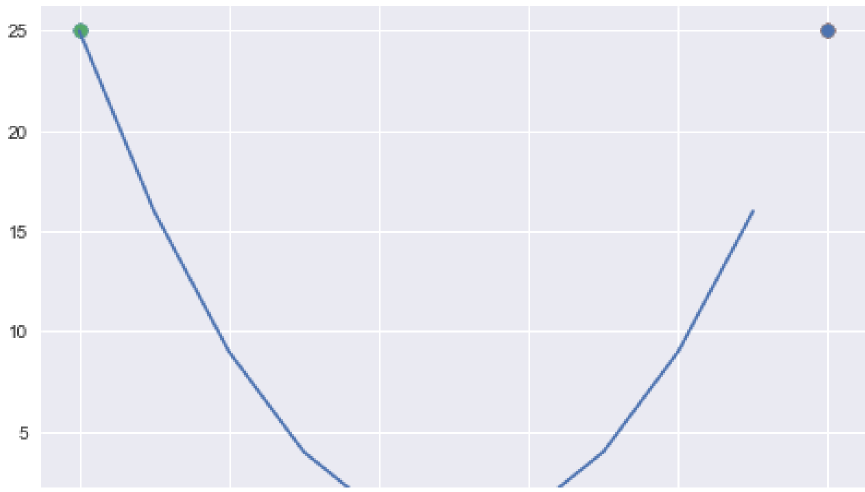
```
[0 1 2 3 4 5 6 7 8 9] [25 16  9  4  1  0  1  4  9 16]
(10,) (10,)
```

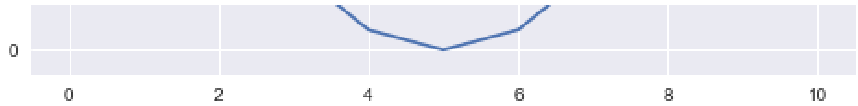


```
[0 1 2 3 4 5 6 7 8 9] [25 16  9  4  1  0  1  4  9 16]
(10,) (10,)
```



```
[0 1 2 3 4 5 6 7 8 9] [25 16  9  4  1  0  1  4  9 16]
(10,) (10,)
```





In [ ]:

```

```

In [ ]:

```

```

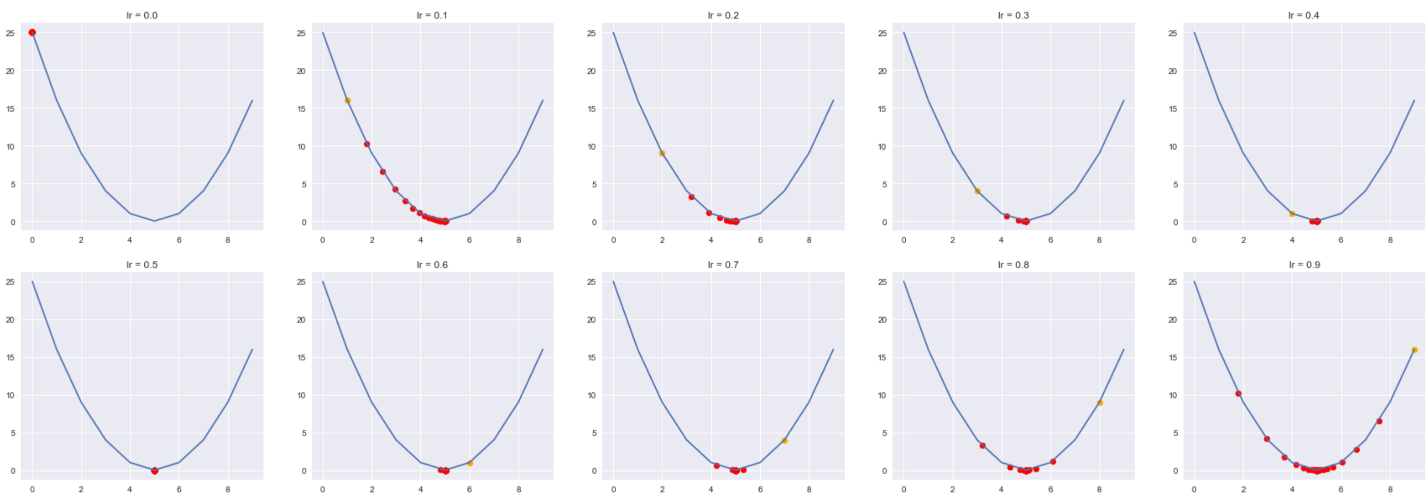
In [ ]:

```

```

Plot showing behaviour of Gradient Descent for different learning rates

In [41]:

```
i = 1
plt.figure(figsize=(30, 10))

for j in range(0, 10, 1):

    X = np.arange(10)
    Y = (X - 5) ** 2

    x = 0
    lr = j / 10
    errors = []
    plt.subplot(2, 5, j + 1)
    plt.title('lr = ' + str(lr))
    plt.plot(X, Y)

    # Take 50 steps in downhill direction
    for i in range(50):
        grad = 2 * (x - 5)
        x = x - lr * grad
        y = (x - 5) ** 2
        errors.append(y)
        if i == 0:
            plt.scatter(x, y, color='Orange')
        else:
            plt.scatter(x, y, color='Red')


    i += 1
plt.show()
```



In [37]:

In [ ]:

```

```