In [1]:
```
print('Hello World')
```

```
Hello World
```

Things to try:

1. Copy the print cell to a new cell below, remove the parenthesis, and run again. What heppened? Why?

In [3]:
```
print'Hello World'
```

```
  File "/tmp/ipykernel_3868/2588146850.py", line 1
    print'Hello World'
          ^
SyntaxError: invalid syntax
```

We get an error here because in python 3 you need to enclose all print statements in parenthesis. Print in python is a function so you need to enclose it in parenthesis just like you would with any other python function. However, in python 2, print is not a function but a statement. So if you are running python 2, parenthesis would not be required and it would print without an invalid syntax error.

2. Write a for loop that prints numbers 0 to 9. Then do the same with a while loop.

In [7]:
```
import numpy as np

for i in range(0,10):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [9]:
```
x = 1
while x < 10:
    print(x)
    x +=1
```

```
1
2
3
4
5
6
7
8
9
```

3. Calculate 2^{x} with 2**(0.5), 2**(1./2), 2**(1/2), 2**(1//2). What is the difference between them?

In [10]:
```
2**(0.5)
```

Out[10]:
```
1.4142135623730951
```

In [11]:
```
2**(1./2)
```

Out[11]:
```
1.4142135623730951
```

In [12]:
```
2**(1/2)
```

Out[12]:
```
1.4142135623730951
```

In [13]:
```
2**(1//2)
```

Out[13]:
```
1
```

Here 0.5, 1./2, and 1/2 are float and will return the same values. But 1//2 is an integer and wont return a decimal place like the rest. The // operator is used to return the nearest integer which is why we see 1 instead of 2.

4. Run program 1.1 for free fall with different initial velocities, interpret the results.

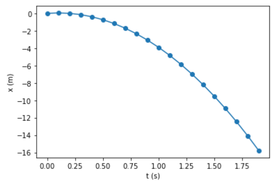
In [24]:
```
#
# Program 1.1: Motion with constant acceleration (motion.py)
# J Wang, Computational modeling and visualization with Python
#

import matplotlib.pyplot as plt # get matplotlib plot functions
import sys

#if (sys.version_info[0] < 3):
#    a, v0 = input('enter a, v0 (eg 1.0, 0.0) : ')   # read input 2.xx
#else:
#    a, v0 = eval(input('enter a, v0 (eg 1.0, 0.0) : '))   # 3.xx
t, h, n,a,v0 = 0.0, 0.1, 20, -9.8, 1          # init time, step size, number of steps
ta, xa = [], []                 # time and position arrays for plotting

for i in range(n):              # loop for n steps
    ta.append(t)                # record time and position
    xa.append(v0*t + a*t*t/2.0)
    t = t + h                   # update time

plt.figure()                    # start a figure; no indent->end of loop
plt.plot(ta, xa, '-o')          # plot data
plt.xlabel('t (s)')             # add labels
plt.ylabel('x (m)')
plt.show()                      # show figure
```

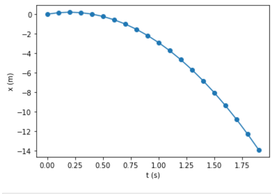


In [25]:
```
import matplotlib.pyplot as plt # get matplotlib plot functions
import sys

#if (sys.version_info[0] < 3):
#    a, v0 = input('enter a, v0 (eg 1.0, 0.0) : ')   # read input 2.xx
#else:
#    a, v0 = eval(input('enter a, v0 (eg 1.0, 0.0) : '))   # 3.xx
t, h, n,a,v0 = 0.0, 0.1, 20, -9.8, 2          # init time, step size, number of steps
ta, xa = [], []                 # time and position arrays for plotting

for i in range(n):              # loop for n steps
    ta.append(t)                # record time and position
    xa.append(v0*t + a*t*t/2.0)
    t = t + h                   # update time

plt.figure()                    # start a figure; no indent->end of loop
plt.plot(ta, xa, '-o')          # plot data
plt.xlabel('t (s)')             # add labels
plt.ylabel('x (m)')
plt.show()
```

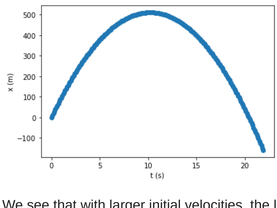


In [29]:
```
import matplotlib.pyplot as plt # get matplotlib plot functions
import sys

#if (sys.version_info[0] < 3):
#    a, v0 = input('enter a, v0 (eg 1.0, 0.0) : ')   # read input 2.xx
#else:
#    a, v0 = eval(input('enter a, v0 (eg 1.0, 0.0) : '))   # 3.xx
t, h, n,a,v0 = 0.0, 0.1, 220, -9.8, 100         # init time, step size, number of steps
ta, xa = [], []                 # time and position arrays for plotting

for i in range(n):              # loop for n steps
    ta.append(t)                # record time and position
    xa.append(v0*t + a*t*t/2.0)
    t = t + h                   # update time

plt.figure()                    # start a figure; no indent->end of loop
plt.plot(ta, xa, '-o')          # plot data
plt.xlabel('t (s)')             # add labels
plt.ylabel('x (m)')
plt.show()
```



We see that with larger initial velocities, the longer the distance the particle travels. We also notice that the particles mass is not used in this code example. That is because free fall motion does not depend on the particles mass. Physically, in flat spacetime, we would see the particle stop at 0. But mathematically, if we include more time steps, we see the particle move past the zero mark which is ok from a math standpoint. This plot shows us a particle moving straight up and down. How high the particle traveled upwards and the time it took to get back to the starting point.

In [ ]:
```
```