¿Cómo calculamos movimientos con la compu?

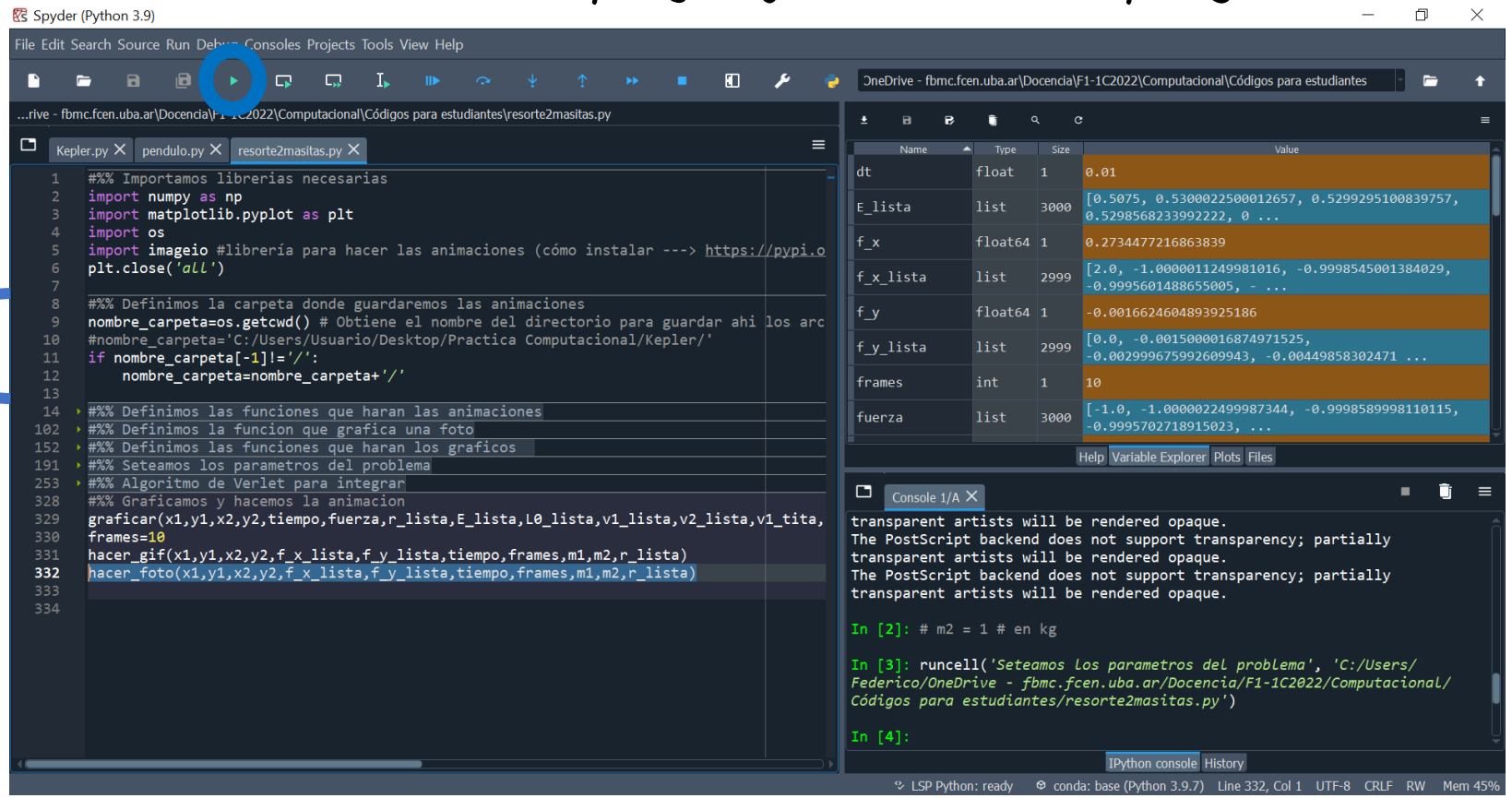
(Integración numérica)

1) Un poco de programación

Con el botón de "play" ejecuto todo el programa

Shift+Enter ejecuto un bloque

F9 ejecuto una selección

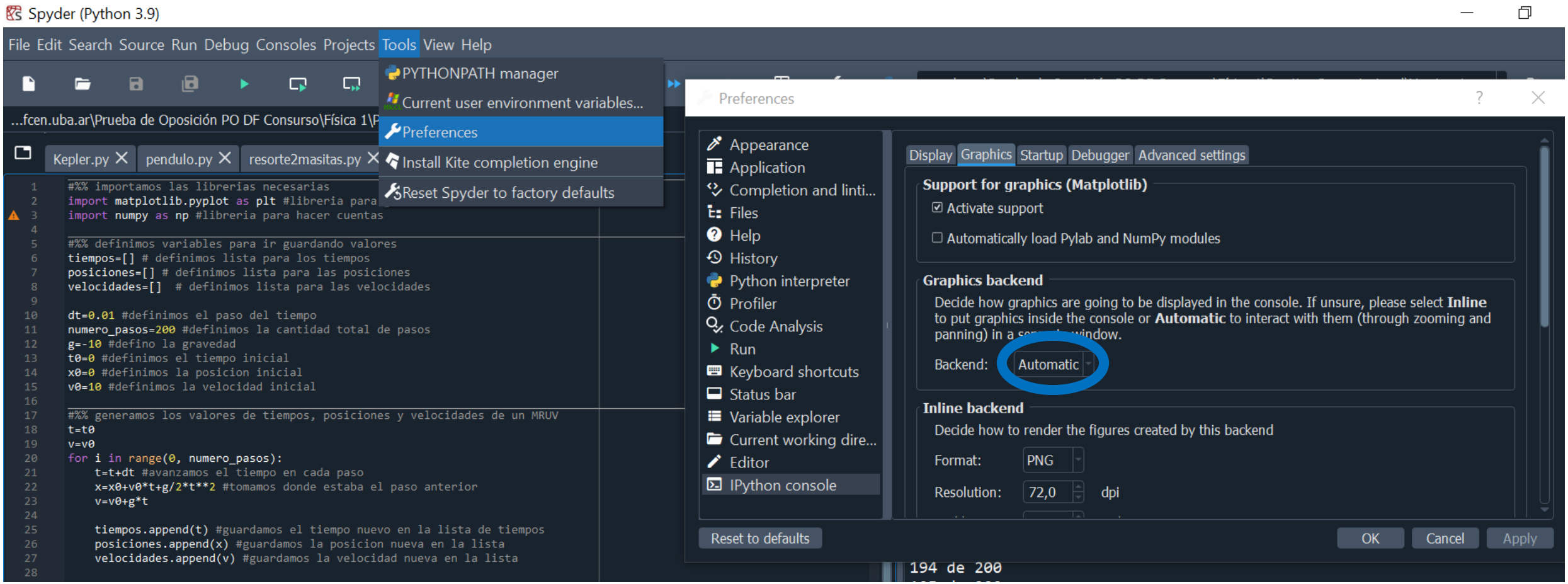


El explorador de variables muestra el valor de variables que voy generando

Acá el código o script, la secuencia de órdenes que le doy a la compu

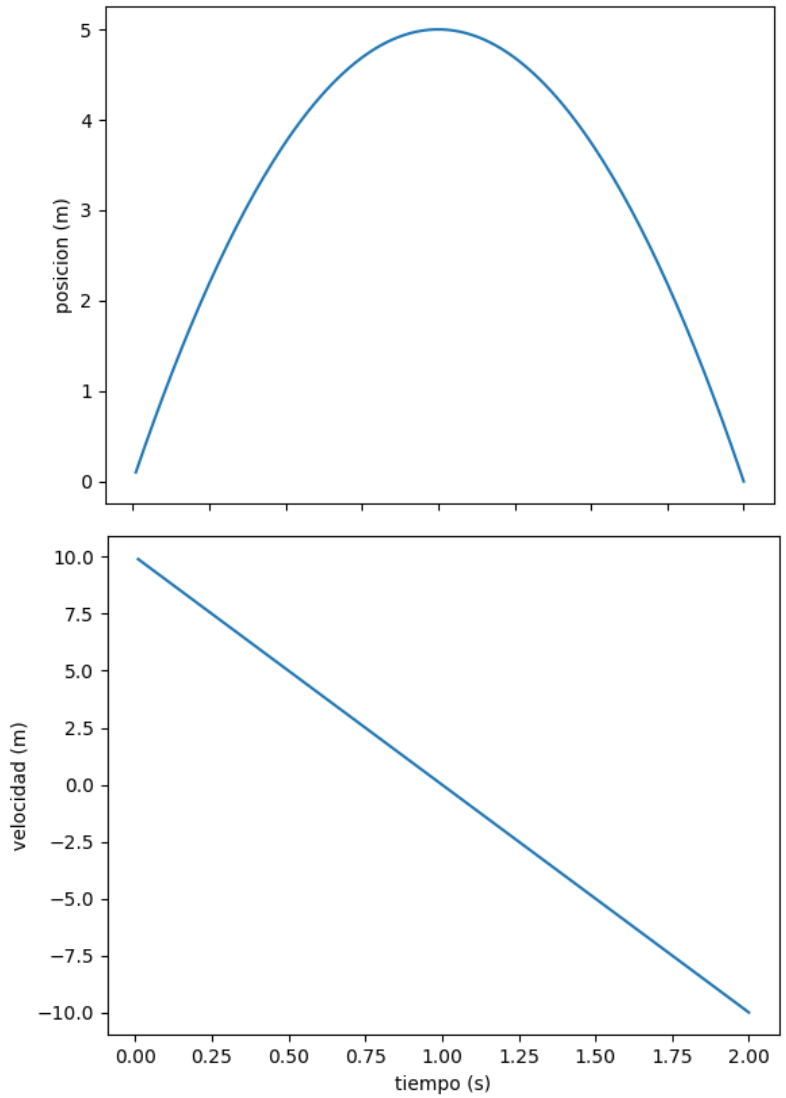
Esta es la consola donde también puedo ejectuar órdenes de a una

1) Un poco de programación



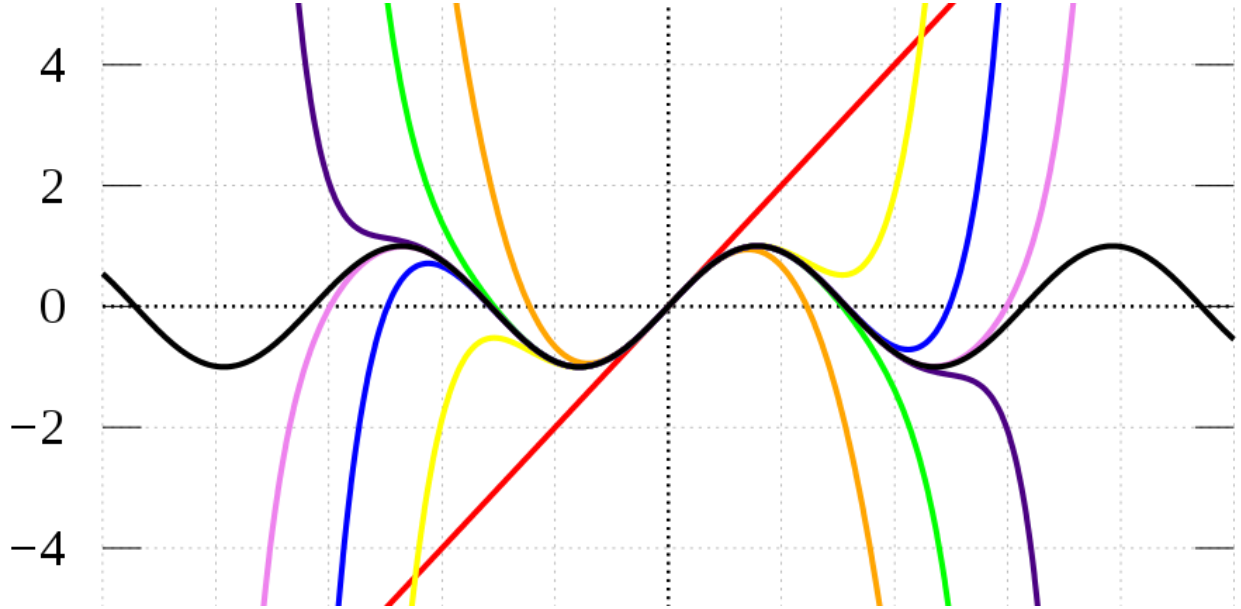
1) Un poco de programación

```python
#%% importamos las librerias necesarias
import matplotlib.pyplot as plt #libreria para graficar
import numpy as np #libreria para hacer cuentas

#%% definimos variables para ir guardando valores
tiempos=[] # definimos lista para los tiempos
posiciones=[] # definimos lista para las posiciones
velocidades=[]  # definimos lista para las velocidades

dt=0.01 #definimos el paso del tiempo
numero_pasos=200 #definimos la cantidad total de pasos
g=-10 #defino la gravedad
t0=0 #definimos el tiempo inicial
x0=0 #definimos la posicion inicial
v0=10 #definimos la velocidad inicial

#%% generamos los valores de tiempos, posiciones y velocidades de un MRUV
t=t0
v=v0
for i in range(0, numero_pasos):
    t=t+dt #avanzamos el tiempo en cada paso
    x=x0+v0*t+g/2*t**2 #tomamos donde estaba el paso anterior
    v=v0+g*t

    tiempos.append(t) #guardamos el tiempo nuevo en la lista de tiempos
    posiciones.append(x) #guardamos la posicion nueva en la lista
    velocidades.append(v) #guardamos la velocidad nueva en la lista

    print(str(i)+' de '+str(numero_pasos)) #hacemos que imprima el valor de i en pantalla

#%% graficamos
#Grafico la posición vs tiempo
plt.figure() #abrimos una nueva figura para graficar
plt.plot(tiempos, posiciones) #hago el grafico
plt.xlabel('tiempo (s)') #nombre del eje X
plt.ylabel('posicion (m)') #nombre del eje Y
plt.show() #muestra la figura

#Grafico la posición vs tiempo
plt.figure() #abrimos una nueva figura para graficar
plt.plot(tiempos, velocidades) #hago el grafico
plt.xlabel('tiempo (s)') #nombre del eje X
plt.ylabel('velocidad (m)') #nombre del eje Y
plt.show() #muestra la figura
```



2) Un poco de matemática

Teorema de Taylor: puedo aproximar f(x) alrededor de un x_o mediante un polinomio



$$r(t+dt) = r(t) + \dot{r}(t)dt + \frac{\ddot{r}(t)}{2!}dt^2 + \frac{\dddot{r}(t)}{3!}dt^3 + \cdots$$

$$r(t+dt) \approx r(t) + \dot{r}(t)dt + \frac{\ddot{r}(t)}{2}dt^2 + \frac{\dddot{r}(t)}{3!}dt^3$$

$$r(t-dt) \approx r(t) - \dot{r}(t)dt + \frac{\ddot{r}(t)}{2}dt^2 - \frac{\dddot{r}(t)}{3!}dt^3$$

+

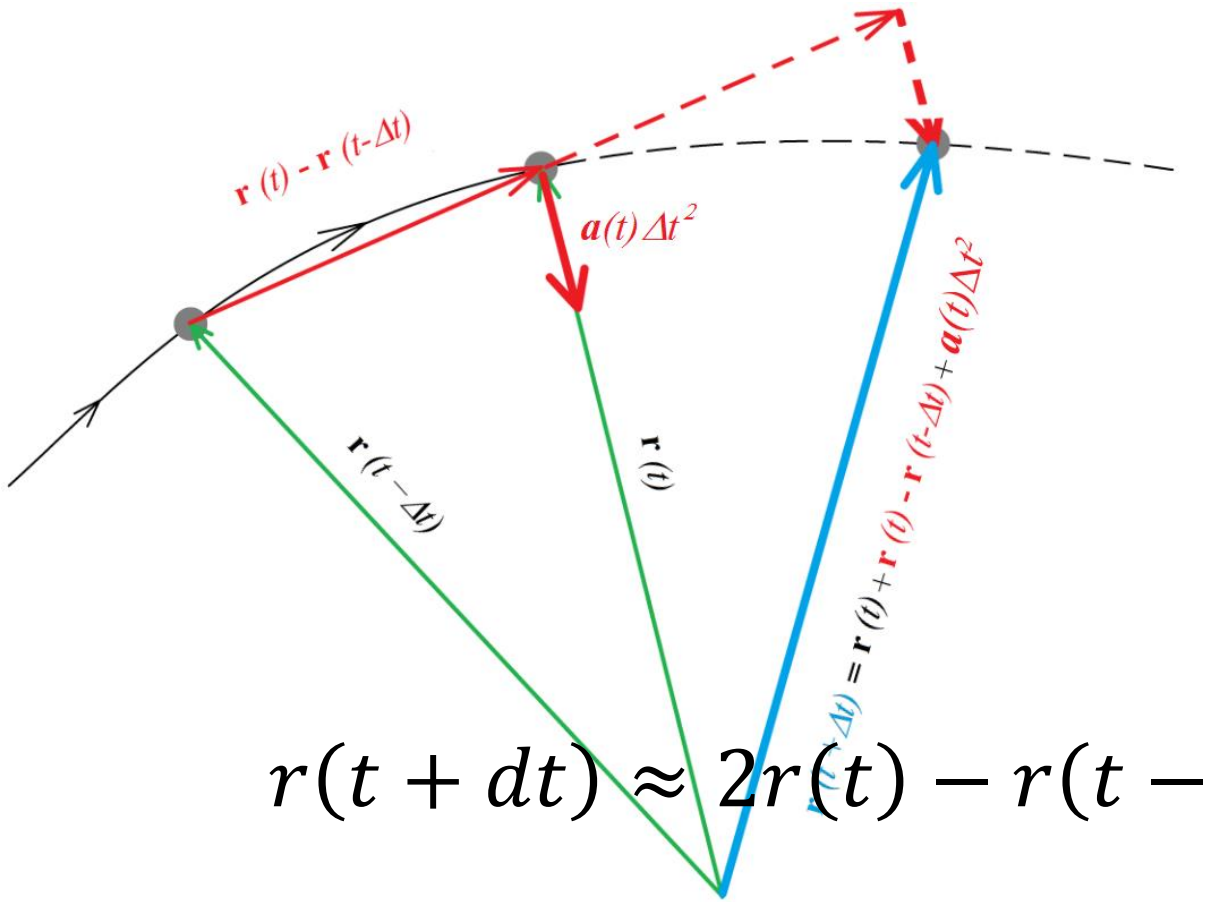
$$r(t+dt) + r(t-dt) \approx 2r(t) + \ddot{r}(t)dt^2$$

$$r(t+dt) \approx 2r(t) - r(t-dt) + \ddot{r}(t)dt^2$$

$$r(t+dt) \approx 2r(t) - r(t-dt) + \frac{\sum F(r(t),t)}{m}dt^2$$

2) Un poco de matemática

$$r(t+dt) \approx 2r(t) - r(t-dt) + \frac{\sum F(r(t),t)}{m} dt^2$$



Sabiendo dónde está el sistema ahora $r(t)$, de dónde vino $r(t-dt)$ y las fuerzas que recibe $\sum F(r(t),t)$, podemos predecir dónde estará $r(t+dt)$

La hipótesis es que el dt es lo suficientemente chico como para asumir que la $\sum F(r(t),t)$ es cte.

$$r(t+dt) \approx 2r(t) - r(t-dt) + \frac{\sum F(r(t),t)}{m} dt^2$$

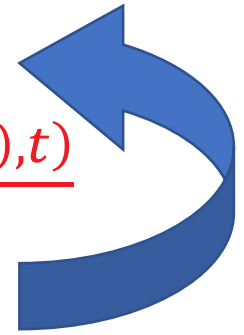
3) Y que se haga la física

$$r(t+dt) \approx 2r(t) - r(t-dt) + \frac{\sum F(r(t),t)}{m} dt^2$$

Algoritmo de Verlet:

1) Elegimos una posición $r(t)$
2) Calculamos la fuerza y la aceleración $\frac{\sum F(r(t),t)}{m}$
3) Calculamos la posición siguiente $r(t+dt)$

Loop o ciclo



$t \leftarrow t + dt$

es **asignación**, no igualdad

La idea es tomar un dt pequeño donde valga la aproximación de Taylor y que $F(r(t),t) = cte$

Algoritmo de Verlet:

1) Elegimos una posición $r(t)$
2) Calculamos la fuerza y la aceleración $\frac{\sum F(r(t),t)}{m}$
3) Calculamos la posición siguiente $r(t+dt)$

$$r(t+dt) \approx 2r(t) - r(t-dt) + \frac{\sum F(r(t),t)}{m} dt^2$$

```
110  #%% Algoritmo de Verlet
111  pasos = 1000 #número de pasos de la simulación
112  dt=0.01
113  t0=0
114
115  tita=[]
116  tita.append(posicion_inicial)
117  tita.append(posicion_inicial+velocidad_inicial*dt)
118
119  tiempo = [t0-dt, t0]
120  velocidad_tangencial = [(tita[1]-tita[0])/dt]
121  aceleracion=[-g/L*np.sin(tita[0]),-g/L*np.sin(tita[1])-viscocida
122  tension=[g*np.cos(tita[0])+L*velocidad_tangencial[0]**2]
123
124  tiempo_actual=tiempo[1]
125  for i in range(1, pasos - 1):
126      #actualizo las posiciones actual y previa
127      tita_actual = tita[i]
128      tita_prev = tita[i-1]
129      #actualizo la velocidad actual
130      velocidad_tangencial.append((tita_actual-tita_prev)/dt)
131      #calculo la aceleración actual y la guardo
132      a_tita = -g/L*np.sin(tita[i])-viscocidad*velocidad_tangencia
133      aceleracion.append(a_tita)
134      #calculo la posición nueva y la guardo
135      tita_nueva = 2 * tita_actual - tita_prev + a_tita * dt**2
136      tita.append(tita_nueva)
137      #calculo la tension y la guardo
138      tension.append(g*np.cos(tita_actual)+L*velocidad_tangencial[
139      #actualizo el tiempo y lo guardo
140      tiempo_actual=tiempo_actual+dt
141      tiempo.append(tiempo_actual)
```

¿Qué hacemos?

Hay 3 programas:

1) Resorte2masitas	Jueguen
2) Péndulo (barra)	Resuelvan la guía
3) Kepler (órbita de la tierra)	Resuelvan la guía

¿Qué hacemos?

Hay 3 programas:

1) Resorte2masitas — Jueguen
2) Péndulo (barra) — Resuelvan la guía
3) Kepler (órbita de la tierra) — Resuelvan la guía

A jugar