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

```
dias=np.random.randint(1,366,size=20)
print(dias)
```

```
[294 274  86 221 124 239 136 307  18 337 325 302 295  56 104 176 325 182
 240  82]
```

```
for i in range(0,20):
     n1=dias[i]
     v=dias[i+1:]
     for j in range(0,len(v)):
         if n1==v[j]:
              print(dias)
```

```
[294 274  86 221 124 239 136 307  18 337 325 302 295  56 104 176 325 182
 240  82]
```

```
def generaDias(nEstudiantes):
    dias=np.random.randint(1,366,size=nEstudiantes)
    op=False
    for i in range(0,nEstudiantes):
        n1=dias[i]
        v=dias[i+1:]
        for j in range(0,len(v)):
            if n1==v[j]:
                op=True
    return op
```

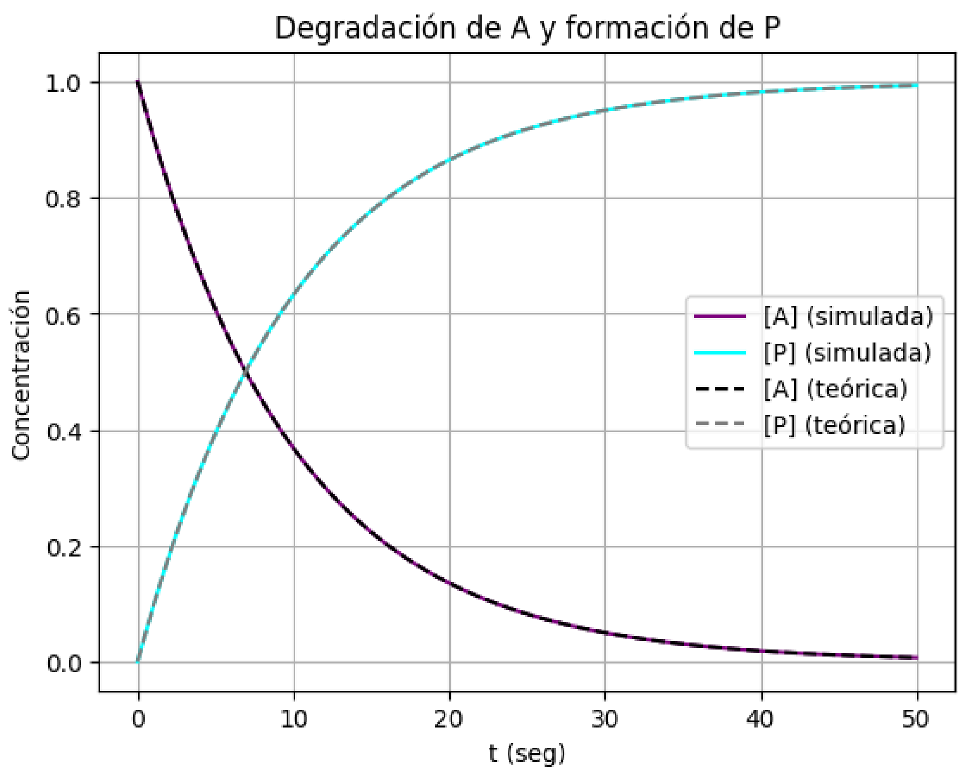
```
n=0
for i in range (1000):
     if generaDias(20)==True:
         n+=1
print(n/1000*100)
```

```
42.8
```

Reacción Química A__>P k

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

A = 1.0
P = 0.0
k = 0.1
t = 0
dt = 0.01
tt = []
AA = []
PP = []

while t < 50:
```

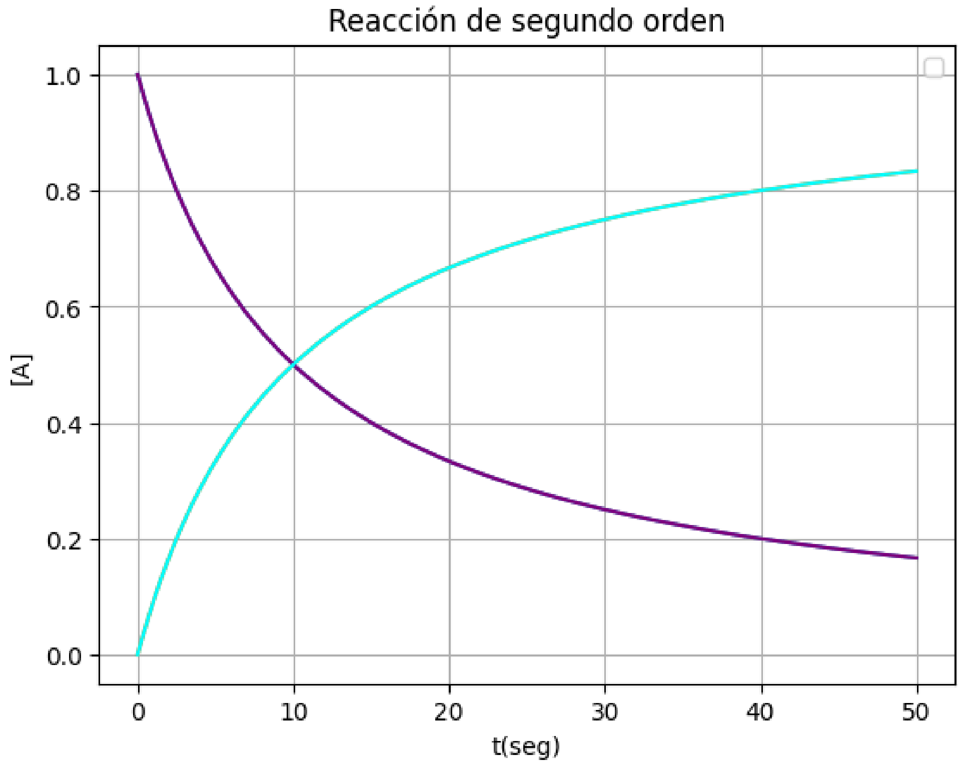
```
while t < 50:
    tt.append(t)
    AA.append(A)
    PP.append(P)
    dA = -k * A * dt
    dP = k * A * dt
    A = A + dA
    P = P + dP
    t += dt

tt = np.array(tt)
AA = np.array(AA)
PP = np.array(PP)

# Gráficas teóricas (opcional)
x = np.linspace(0, 50, 1000)
y = np.exp(-k * x)
y2 = 1 - y

# Gráficas
plt.plot(tt, AA, color="purple", label="[A] (simulada)")
plt.plot(tt, PP, color="cyan", label="[P] (simulada)")
plt.plot(x, y, "--", color="black", label="[A] (teórica)")
plt.plot(x, y2, "--", color="gray", label="[P] (teórica)")

plt.xlabel("t (seg)")
plt.ylabel("Concentración")
plt.title("Degradación de A y formación de P")
plt.legend()
plt.grid(True)
plt.show()
```



A+A - - > P

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

A=1.0
P=0.0
k=0.1
t=0
dt=0.01
tt=[]
AA=[]
PP=[]
while t<50:
    tt.append(t)
    AA.append(A)
    PP.append(P)
    dA=-k*A*A*dt
    dP=k*A*A*dt
    A=A+dA
    P=P+dP
    t+=dt

tt=np.array(tt)
AA=np.array(AA)
PP=np.array(PP)

Ao=1.0
invAt=k*x+1/Ao
y=1/invAt
plt.plot(tt,AA)
plt.plot(tt,PP)

plt.xlabel("t(seg)")
plt.ylabel("[A]")
plt.plot(tt, AA, color="purple")
plt.plot(tt, PP, color="cyan")
plt.title("Reacción de segundo orden")
plt.legend()
plt.grid(True)
plt.show()
```

```
<ipython-input-47-8667f505de74>:37: UserWarning: No artists with labels found to put in legend
  plt.legend()
```

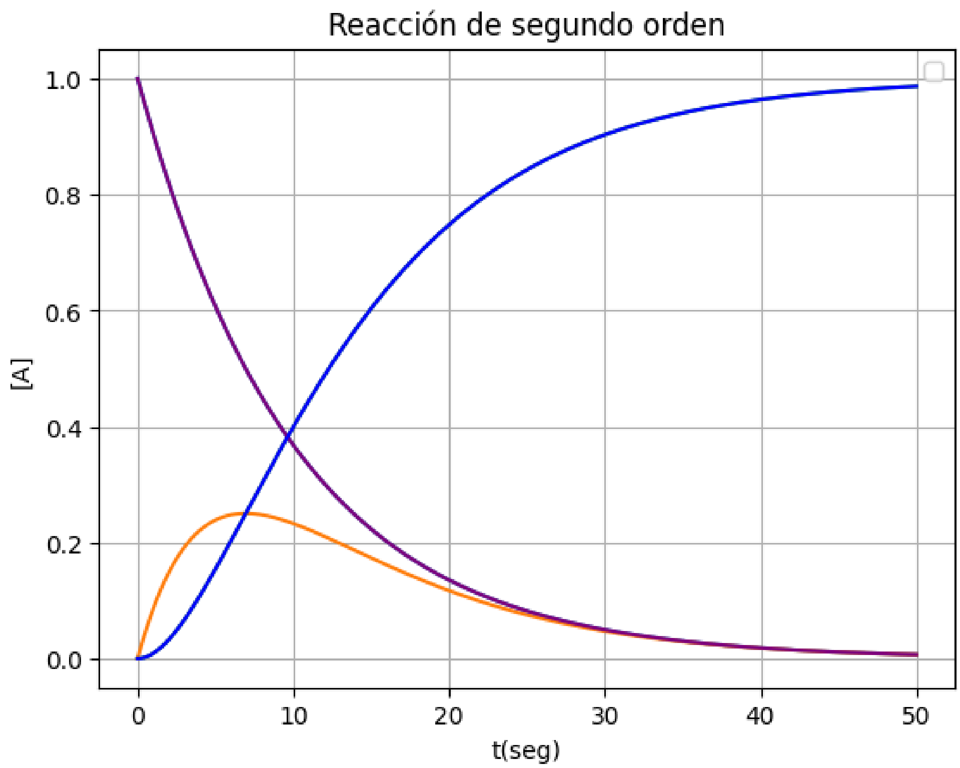


A - - > x k1 vA=-k1*[A] vX=k1*[A] - k2* X - - > P k2

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

A=1.0
P=0.0
X=0
k1=0.1
k2=0.2
t=0
dt=0.01
tt=[]
AA=[]
PP=[]
XX=[]
while t<50:
    tt.append(t)
    AA.append(A)
    PP.append(P)
    XX.append(X)
    dA=-k1*A*dt
    dX=k1*A*dt-k2*X*dt
    dP=k2*X*dt
    A=A+dA
    X=X+dX
    P=P+dP
    t+=dt

tt=np.array(tt)
```

```
AA=np.array(AA)
PP=np.array(PP)
XX=np.array(XX)

plt.plot(tt,AA)
plt.plot(tt,XX)
plt.plot(tt,PP)

plt.xlabel("t(seg)")
plt.ylabel("[A]")
plt.plot(tt, AA, color="purple")
plt.plot(tt, PP, color="blue")
plt.title("Reacción de segundo orden")
plt.legend()
plt.grid(True)
plt.show()
```

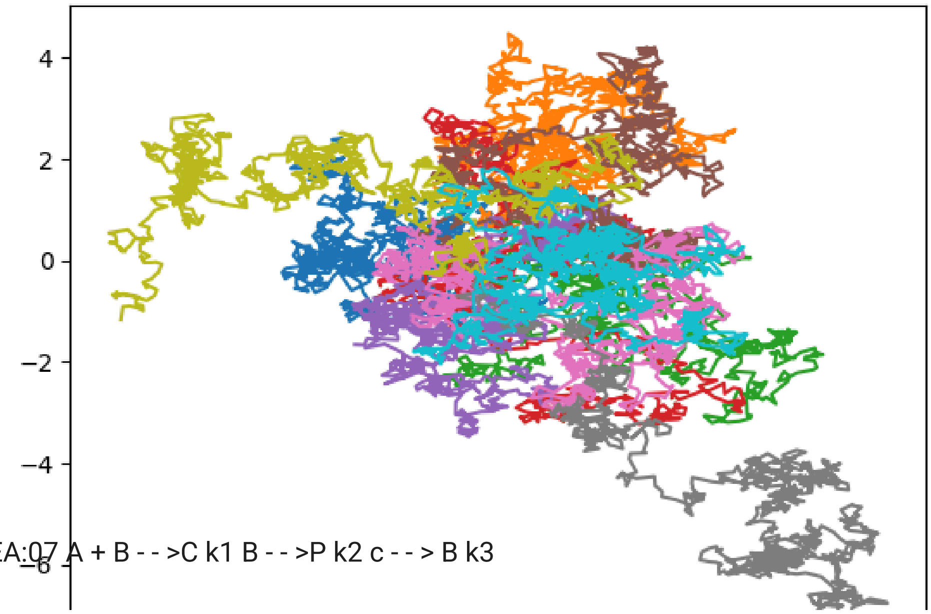
```
<ipython-input-59-27887abb1a9a>:42: UserWarning: No artists with labels found to put in legend
  plt.legend()
```




```
for i in range (0,10):
    dx=np.random.normal(0,0.1,size=1000)
    dy=np.random.normal(0,0.1,size=1000)
    x=np.cumsum(dx)
    y=np.cumsum(dy)
    plt.plot(x, y,)
plt.show
```

```
matplotlib.pyplot.show
def show(*args, **kwargs) -> None
```

```
Display all open figures.

Parameters
----------
block : bool, optional
    Whether to wait for all figures to be closed before returning.
```



TAREA:07 A + B - - >C k1 B - - >P k2 c - - > B k3

```
TAREA: ATG - generar una secuencia de 500 nucleótidos al azar
```

```
TAREA: Contar cuantas A, T, G, C hay en la secuencia qué genera
```

```
  File "<ipython-input-82-c34f08046e43>", line 1
    TAREA: Contar cuantas A, T, G, C hay en la secuencia qué genera
                 ^
SyntaxError: invalid syntax
```