Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencias de la Computación

Clase 11: Sistema de Archivos

Rodrigo Toro Icarte (rntoro@uc.cl)

IIC1103 Introducción a la Programación - Sección 5

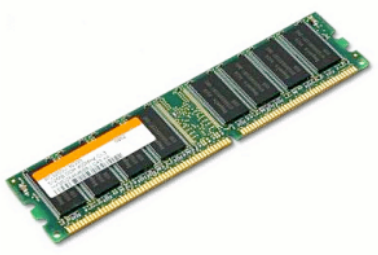
27 de Abril, 2015

Motivación

Hasta ahora, **todos** los programas que hemos creado en el curso tienen **dos problemas**.

Motivación

Hasta ahora, **todos** los programas que hemos creado en el curso tienen **dos problemas**.



Motivación

1) Al cerrar su programa, todo se pierde.

Motivación

1) Al cerrar su programa, todo se pierde.

```
    012 345 678 (c)
    --- --- ---
 0 |987 364 152|
 1 |000 275 498|
 2 |452 918 637|

 3 |893 547 261|
 4 |215 689 374|
 5 |700 132 985|

 6 |620 453 719|
 7 |000 021 506|
 8 |000 000 803|
(f) --- --- ---
```

Motivación

Esto no ocurre en las aplicaciones de la vida real...





Motivación

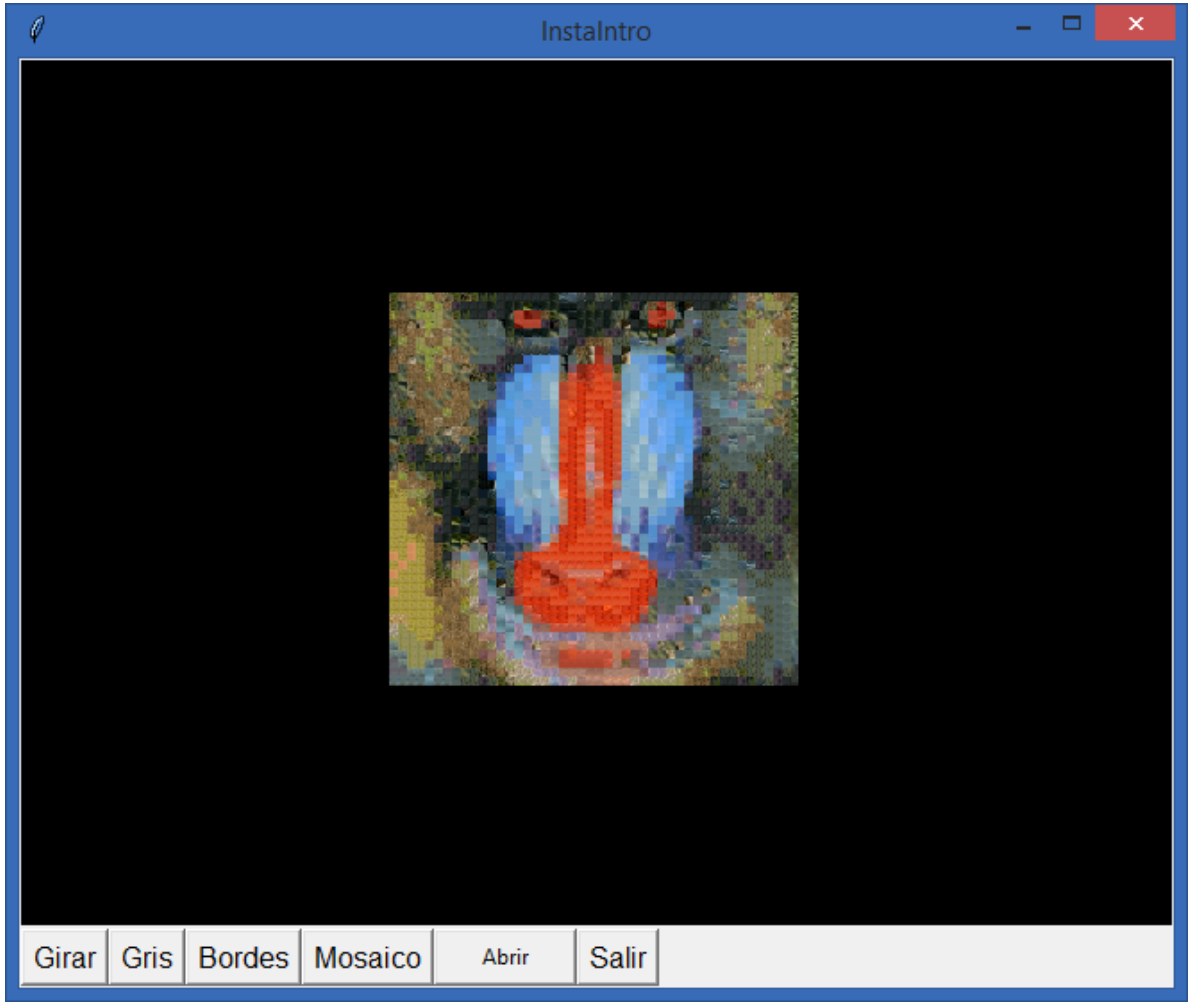
2) Siempre debemos generar resultados desde cero.

"Tenemos memoria, pero no tiempo..."

Motivación

2) Siempre debemos generar resultados desde cero.

"Tenemos memoria, pero no tiempo..."



Motivación

¿Cómo solucionamos estos problemas?

Motivación

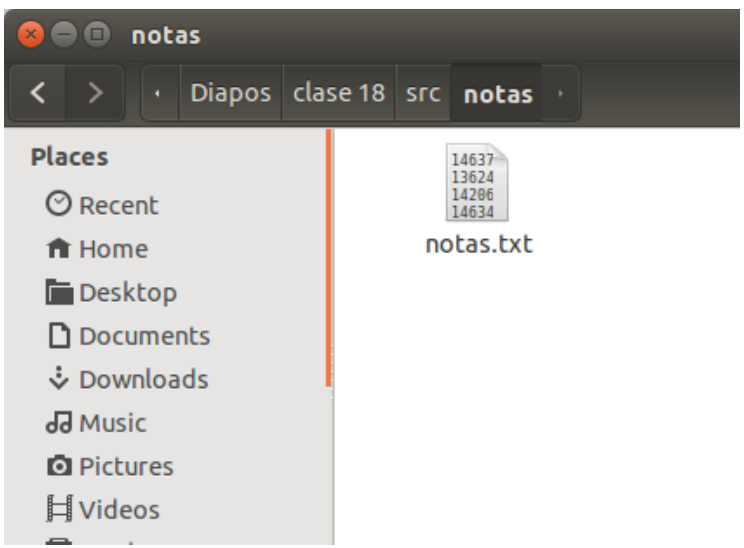
¿Cómo solucionamos estos problemas?

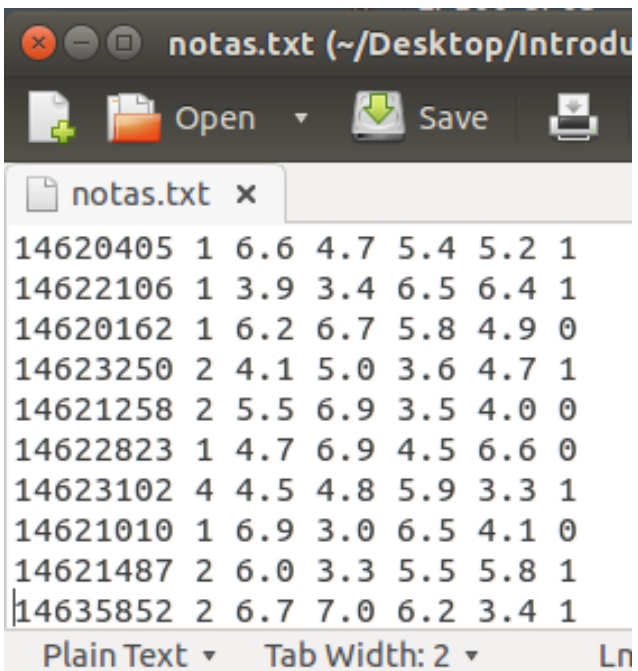
Necesitamos aprender a leer y escribir archivos desde python.

Motivación

¿Cómo solucionamos estos problemas?

Necesitamos aprender a leer y escribir archivos desde python.

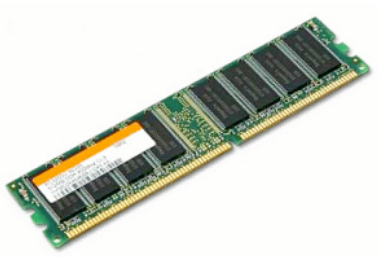




Sistema de archivos

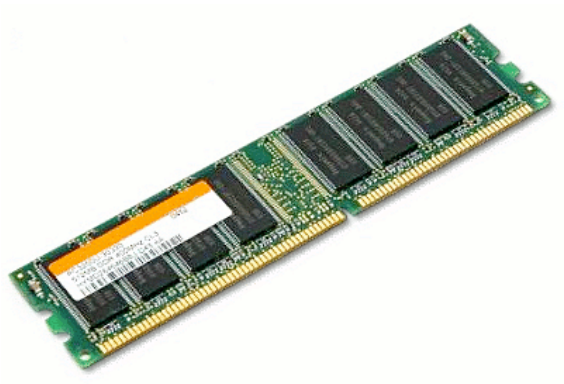
Sistema de archivos

¿Dónde viven los datos en un computador?



Sistema de archivos

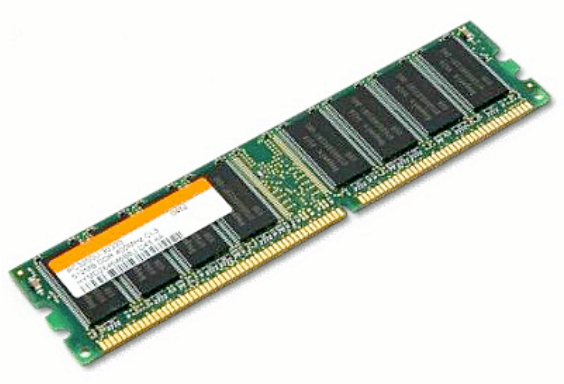
Memoria principal (RAM)



- Funciona en base a corriente.
- Es muy rápida.
- Costosa (1Gb a 8Gb en PC convencional).
- Volátil.

Sistema de archivos

Memoria principal (RAM)



- Funciona en base a corriente.
- Es muy rápida.
- Costosa (1Gb a 8Gb en PC convencional).
- Volátil.

Aquí viven las variables de nuestro programa cuando lo ejecutamos.

Si se cierra el programa o se apaga el PC todo se pierde.

Sistema de archivos

Disco

- Funciona en base a cambios magnéticos en un disco físico.
- Es lenta.
- Barata (200Gb a 1Tb en PC convencional).
- No volátil.

Sistema de archivos

Disco

- Funciona en base a cambios magnéticos en un disco físico.
- Es lenta.
- Barata (200Gb a 1Tb en PC convencional).
- No volátil.

Aquí guardamos las cosas que queremos que perduren:
Documentos, imágenes, códigos python, programas, etc...

Sistema de archivos

Sistema de archivos

Es la forma en que el sistema operativo guarda y organiza su memoria permanente.

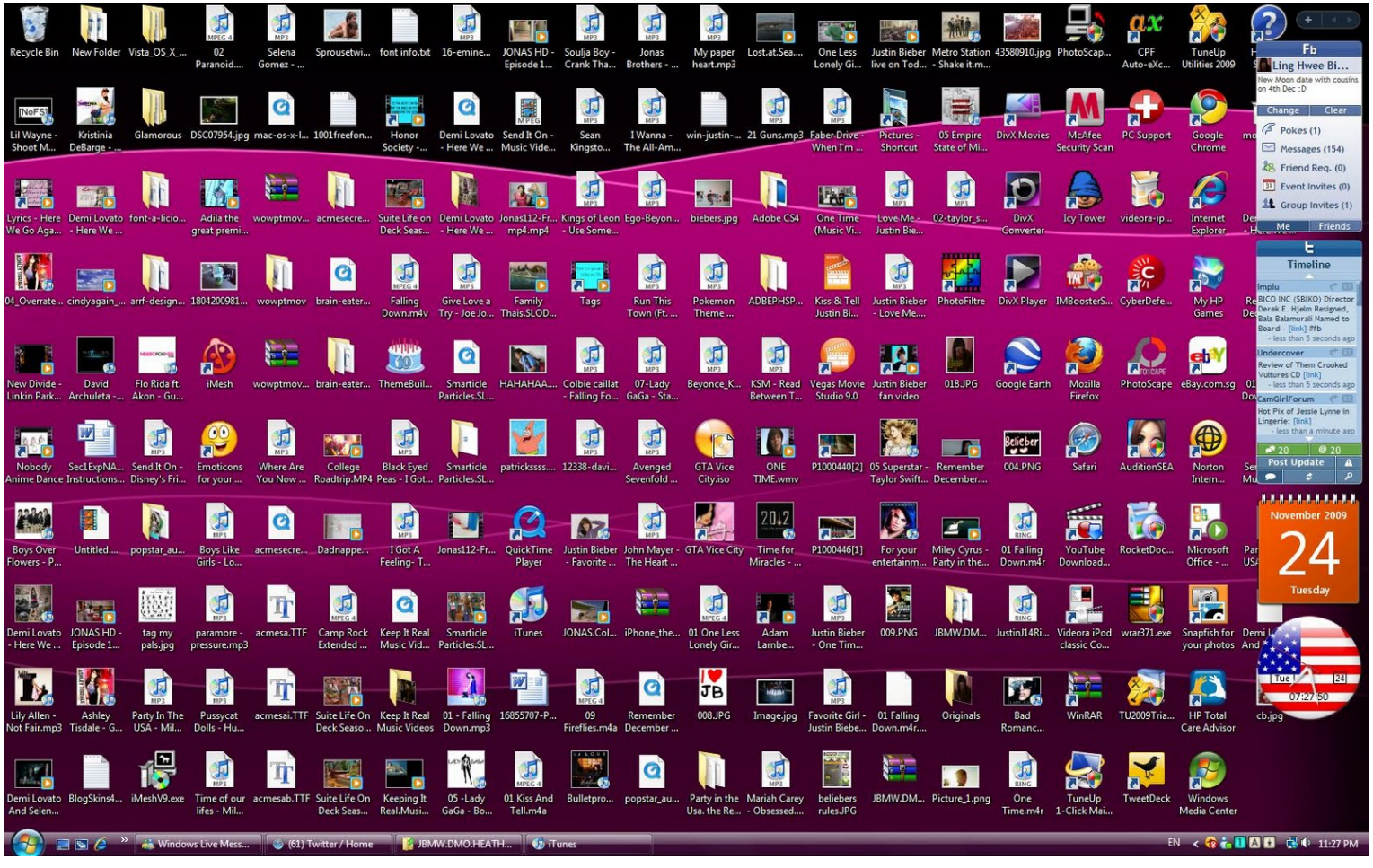
Sistema de archivos

Sistema de archivos

Es la forma en que el sistema operativo guarda y organiza su memoria permanente.

En la prehistoria de las ciencias de la computación los discos solo contenían un tipo de elemento llamado **archivo**.

Sistema de archivos



Sistema de archivos

Problemas:

- Difícil encontrar archivos.
- Colisiones de nombres.

Sistema de archivos

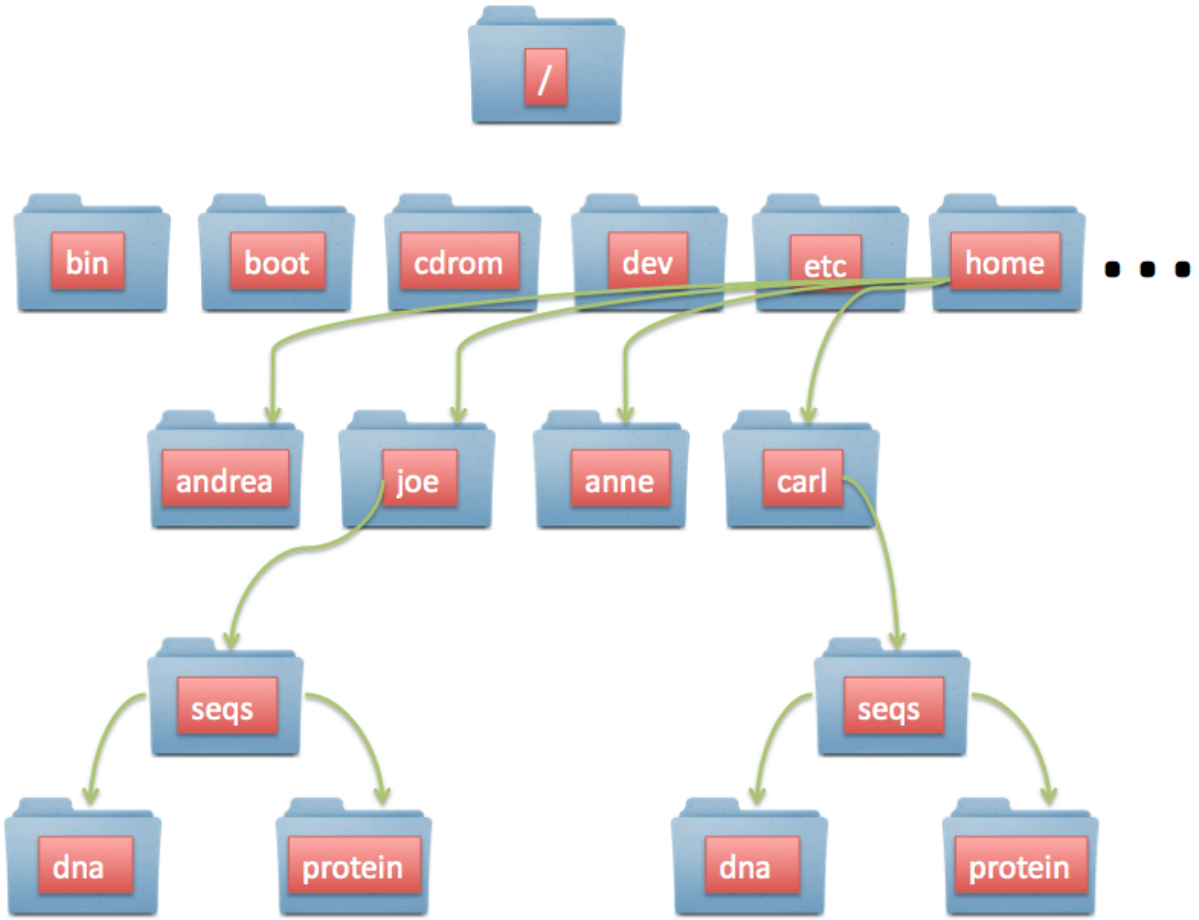
Problemas:

- Difícil encontrar archivos.
- Colisiones de nombres.

Evolución: Agregar carpetas al sistema de archivos.

- Carpetas contienen archivos y subcarpetas.
- Cada archivo pertenece a una carpeta.
- Existe una carpeta inicial llamada **root**.

Sistema de archivos



Sistema de archivos

¿Cuál es la carpeta root?

Sistema de archivos

¿Cuál es la carpeta root?

- Windows → **C:**
- Unix (Mac y Linux) → **/**

Sistema de archivos

¿Cuál es la carpeta root?

- Windows → **C:**
- Unix (Mac y Linux) → /

Ventajas:

- Es más ordenado.
- Podemos tener archivos con el mismo nombre.
- Podemos navegar por jerarquía de carpetas.

Sistema de archivos

¿Cómo navegamos por las carpetas usando python?

Sistema de archivos

¿Cómo navegamos por las carpetas usando python?

Con `os.getcwd()` podemos ver el directorio actual.

```
1 import os
2 # Nos indica el directorio actual
3 print(os.getcwd())
4 # >>> /home/rodrigo/Desktop/
```

Sistema de archivos

¿Cómo navegamos por las carpetas usando python?

Con `os.getcwd()` podemos ver el directorio actual.

```
1 import os
2 # Nos indica el directorio actual
3 print(os.getcwd())
4 # >>> /home/rodrigo/Desktop/
```

Con `os.chdir(path)` nos movemos a otra carpeta.

Sistema de archivos

Path absoluto: Ruta desde carpeta root hasta el archivo o carpeta.

Sistema de archivos

Path absoluto: Ruta desde carpeta root hasta el archivo o carpeta.

Ejemplos:

- /home/rodrigo/Desktop/
- /home/rodrigo/Desktop/notas.txt
- /home/rodrigo/Desktop/Curso
- /home/rodrigo/Desktop/Curso/notas.txt

Sistema de archivos

Path relativo: Ruta desde carpeta actual hasta el archivo o carpeta.

Sistema de archivos

Path relativo: Ruta desde carpeta actual hasta el archivo o carpeta.

Notaciones importantes:

- ./ → Se refiere a la carpeta actual.
- ../ → Se refiere a la carpeta padre.

Sistema de archivos

Path relativo: Ruta desde carpeta actual hasta el archivo o carpeta.

Notaciones importantes:

- ./ → Se refiere a la carpeta actual.
- ../ → Se refiere a la carpeta padre.

Ejemplos:

- ./Documents
- ../notas.txt

Sistema de archivos

```python
import os

# Nos indica el directorio actual
print(os.getcwd())
# >>> /home/rodrigo/Desktop/

# Moverse a la carpeta padre
os.chdir("../")
print(os.getcwd())
# >>> /home/rodrigo/

# Me muevo a una carpeta en forma absoluta
os.chdir("/home/rodrigo/Desktop/Tesis")
print(os.getcwd())
# >>> /home/rodrigo/Desktop/Tesis
```

Sistema de archivos

Otros operadores útiles...

`os.path.exists(p)`: Retorna `True` si `p` existe.
`os.path.isfile(p)`: Retorna `True` si `p` es un archivo.
`os.path.isdir(p)`: Retorna `True` si `p` es una carpeta.

`os.listdir(c)`: Retorna una lista con los elementos de `c`.

`os.makedir(c)`: Crea la carpeta `c`.
`os.remove(p)`: Borra el archivo `p`.
`os.rmdir(c)`: Borra la carpeta `c` (que debe estar vacía).

Sistema de archivos

Ejemplo: Mostrar en forma recursiva la jerarquía de archivos y subcarpetas a partir del directorio actual.

Sistema de archivos

Ejemplo: Mostrar en forma recursiva la jerarquía de archivos y subcarpetas a partir del directorio actual.

```
1 import os
2 def mostrar_archivos(tabs,carpeta):
3   l = os.listdir(carpeta)
4   for p in l:
5     print(tabs + p)
6     if(os.path.isdir(carpeta + "/" + p)):
7       mostrar_archivos("   " + tabs,carpeta+"/"+p)
8
9 mostrar_archivos("-> ",".")
```

Sistema de archivos

Archivos: Son documentos binarios con un nombre y una extensión (notas.txt → **nombre**: notas, **extensión**: .txt).

Sistema de archivos

Archivos: Son documentos binarios con un nombre y una extensión (notas.txt → **nombre**: notas, **extensión**: .txt).

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

/home/rodrigo/Desktop/Curso/notas.txt

Sistema de archivos

Para decodificar el archivo binario existen infinitas opciones.

Sistema de archivos

Para decodificar el archivo binario existen infinitas opciones.

Ejemplos:

- Pasar cada byte (8 bits) a número.
- Pasar cada byte (8 bits) a ASCII.
- Ver los bytes como pixeles en una imagen.
- Etc...

Sistema de archivos

Para decodificar el archivo binario existen infinitas opciones.

Ejemplos:

- Pasar cada byte (8 bits) a número.
- Pasar cada byte (8 bits) a ASCII.
- Ver los bytes como pixeles en una imagen.
- Etc...

La **extensión** nos indica cómo decodificar cada tipo de archivo.

Sistema de archivos

Algunas extensiones comunes:

- .txt → Documento de texto plano.
- .doc → Documento word.
- .jpg → Imágenes.
- .mp3 → Música.

Sistema de archivos

Algunas extensiones comunes:

- .txt → Documento de texto plano.
- .doc → Documento word.
- .jpg → Imágenes.
- .mp3 → Música.

Nosotros nos centraremos en el formato de **texto plano**.

Sistema de archivos

Algunas extensiones comunes:

- .txt → Documento de texto plano.
- .doc → Documento word.
- .jpg → Imágenes.
- .mp3 → Música.

Nosotros nos centraremos en el formato de **texto plano**.

Observación: .py es formato de texto plano.

Lectura de archivos

Lectura de archivos

Motivación: ¿No están chatos del `input()`?

Lectura de archivos

Motivación: ¿No están chatos del `input()`?

Ejemplo: Haga un programa que calcule las notas finales del curso.

$$NF = 0,3 \cdot C + 0,3 \cdot mt + 0,4 \cdot E$$

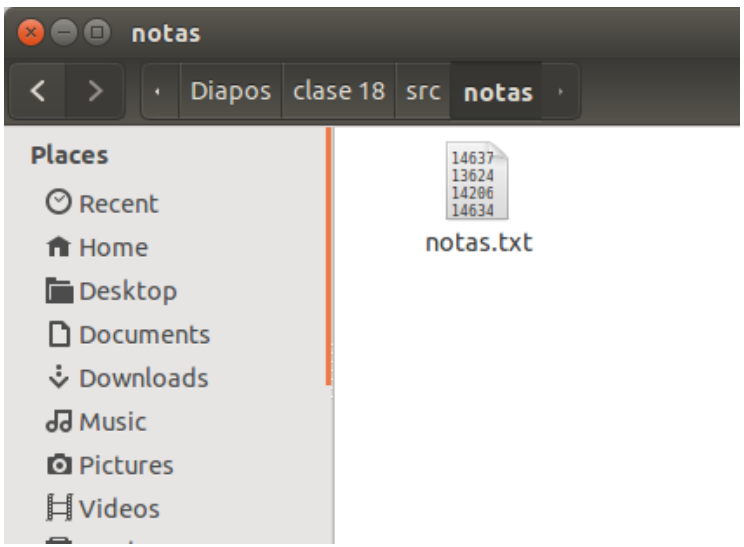
$$C = \frac{1}{3} \sum_{i=1}^{3} c_i + 0,5 \cdot \delta$$

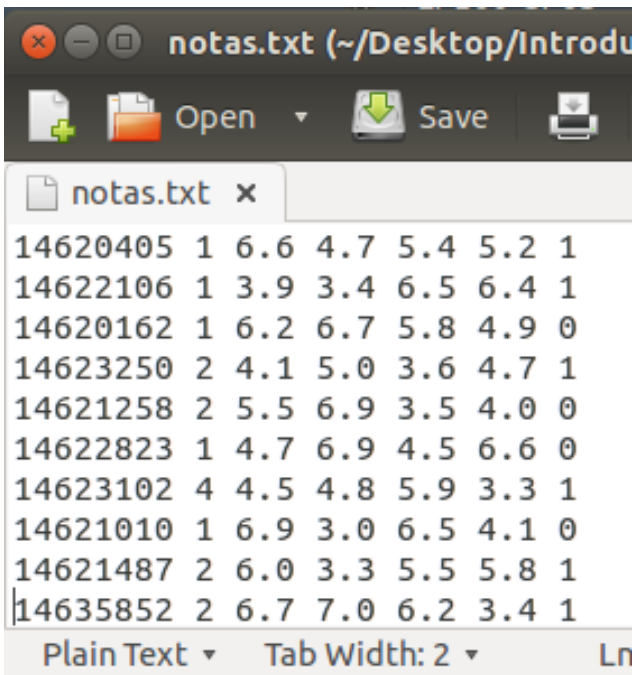
Lectura de archivos

```
1  def calcular_nota_final(n):
2    Bn = int(n[6]); MT = float(n[4]); Ex = float(n[5])
3    C = sum([float(n[1]),float(n[2]),float(n[3])])/3
4    C += int(Bn)*0.5
5    nota_final = min([0.3*C + 0.3*MT + 0.4*Ex,7])
6    if(C<4 or Ex<4): nota_final=min([nota_final,3.9])
7    return nota_final
8
9  suma = 0; total = 0; reprobados = 0
10 while(True):
11   # formado: [Num_alumno C1 C2 C3 MT Ex Bn]
12   n = input('ingrese datos alumno: ').split(' ')
13   if(len(n) != 7): break
14   nf = calcular_nota_final(n)
15   if(nf < 4): reprobados+=1
16   suma += nf; total += 1
17   print(n[0],"%0.1f"%nf)
18 print("promedio",suma/total)
19 print("reprobados",reprobados)
```

Lectura de archivos

Meta: Aprender a leer datos desde archivos de texto.





Lectura de archivos

`f = open(p)`: Función que recibe una **ruta** a un archivo y retorna un objeto del tipo `_io.TextIOWrapper`.

Lectura de archivos

`f = open(p)`: Función que recibe una **ruta** a un archivo y retorna un objeto del tipo `_io.TextIOWrapper`.

`f.readline()`: Método que lee la siguiente línea del archivo.

Lectura de archivos

`f = open(p)`: Función que recibe una **ruta** a un archivo y retorna un objeto del tipo `_io.TextIOWrapper`.

`f.readline()`: Método que lee la siguiente línea del archivo.

`f.close()`: Método que cierra un archivo.

Lectura de archivos

```
1  El amor es un mejor profesor que el deber.
2  Información no es conocimiento.
3  Nunca pierdas la sagrada curiosidad.
4  Si no puedes explicarlo de forma simple, no lo entiendes lo
       suficientemente bien.
5  Todos deben ser respetados como individuos, pero no idolatrados.
6
7  Nunca hagas algo contra tu conciencia incluso si la situación lo
       demanda.
8  Locura: hacer lo mismo una y otra vez, esperando diferentes
       resultados.
9  Un hombre debe buscar lo que es, no lo que piensa que es.
10 Una persona que nunca ha cometido un error nunca ha intentado algo
       nuevo.
11 Aprende del ayer, vive del hoy, espera del mañana.
12
13 Todo debe ser tan simple como es, pero no más sencillo.
14 La lógica te lleva del punto A al B. La imaginación te llevará donde
       sea.
```

./Einstein.txt

Lectura de archivos

```
1  El amor es un mejor profesor que el deber.
2  Información no es conocimiento.
3  Nunca pierdas la sagrada curiosidad.
4  Si no puedes explicarlo de forma simple, no lo entiendes lo
       suficientemente bien.
5  Todos deben ser respetados como individuos, pero no idolatrados.
6
7  Nunca hagas algo contra tu conciencia incluso si la situación lo
       demanda.
8  Locura: hacer lo mismo una y otra vez, esperando diferentes
       resultados.
9  Un hombre debe buscar lo que es, no lo que piensa que es.
10 Una persona que nunca ha cometido un error nunca ha intentado algo
       nuevo.
11 Aprende del ayer, vive del hoy, espera del mañana.
12
13 Todo debe ser tan simple como es, pero no más sencillo.
14 La lógica te lleva del punto A al B. La imaginación te llevará donde
       sea.
```

./Einstein.txt

¿Cómo leemos esto desde python?

Lectura de archivos

```
1  f = open('./Einstein.txt')
2
3  l = f.readline()
4  # "El amor es un mejor profesor que el deber.\n"
5  l = f.readline()
6  # "Información no es conocimiento.\n"
7  l = f.readline()
8  # "Nunca pierdas la sagrada curiosidad.\n"
9
10 f.close()
```

Lectura de archivos

```
1  f = open('./Einstein.txt')
2
3  l = f.readline()
4  # "El amor es un mejor profesor que el deber.\n"
5  l = f.readline()
6  # "Información no es conocimiento.\n"
7  l = f.readline()
8  # "Nunca pierdas la sagrada curiosidad.\n"
9
10 f.close()
```

Observaciones readline():

- Lee hasta el siguiente '\n'.
- Avanza automáticamente a la siguiente línea.
- La línea retornada incluye un '\n' al final.
- Final del archivo es un string vacío ''.

Lectura de archivos

Ejemplo: Leer y mostrar en consola el archivo ./Einstein.txt

```
1 f = open('./Einstein.txt')
2
3 l = f.readline()
4 while(l != ''):       # línea vacía -> EOF
5   print(l.rstrip())
6   l = f.readline()    # leemos siguiente línea
7
8 f.close()             # siempre cierren el archivo!
```

Lectura de archivos

Ejemplo: Leer y mostrar en consola el archivo ./Einstein.txt

```
1 f = open('./Einstein.txt')
2
3 l = f.readline()
4 while(l != ''):       # línea vacía -> EOF
5   print(l.rstrip())
6   l = f.readline()    # leemos siguiente línea
7
8 f.close()             # siempre cierren el archivo!
```

Observaciones:

- Con `rstrip()` eliminamos '\n' al final de cada línea.
- Una línea vacía será '\n', no ''.

Lectura de archivos

Formas alternativas de leer un archivo:

Lectura de archivos

Formas alternativas de leer un archivo:

`for` sobre las líneas del archivo.

```
1 f = open('./Einstein.txt')
2 for linea in f:
3   print(linea.rstrip())
4 f.close()
```

Lectura de archivos

Formas alternativas de leer un archivo:

`for` sobre las líneas del archivo.

```
1 f = open('./Einstein.txt')
2 for linea in f:
3   print(linea.rstrip())
4 f.close()
```

`f.readlines()` retorna una lista con las líneas.

```
1 f = open('./Einstein.txt')
2 lineas = f.readlines()
3 print(lineas)
4 f.close()
```

Lectura de archivos

Archivos separados por comas

Lectura de archivos

Archivos separados por comas

Muchas veces necesitaremos leer tablas a partir de un archivo.

Lectura de archivos

Archivos separados por comas

Muchas veces necesitaremos leer tablas a partir de un archivo.

Ejemplo: Tabla con las notas de cada estudiante.

Lectura de archivos

Archivos separados por comas

Muchas veces necesitaremos leer tablas a partir de un archivo.

Ejemplo: Tabla con las notas de cada estudiante.

Consejo: Definan un caracter que separe los elementos de la tabla, por ejemplo `" "`, `";"`, `","`, `"\t"`.

Lectura de archivos

Ejemplos

notas.txt

```
14620405 1 6.6 4.7 5.4 5.2 1
14622106 1 3.9 3.4 6.5 6.4 1
14620162 1 6.2 6.7 5.8 4.9 0
14623250 2 4.1 5.0 3.6 4.7 1
14621258 2 5.5 6.9 3.5 4.0 0
```

estudiantes.txt

```
Juan;Águila;14000000;6.5;7.0;6.7
Aldo;Verri;14000001;3.0;2.7;3.8
María;Pinto;14000002;5.7;7.0;6.2
Rodrigo;Toro;14000003;1.0;1.0;1.0
```

Lectura de archivos

Ventaja

Leer este tipo de archivos y separar sus atributos es trivial.

Lectura de archivos

Ventaja

Leer este tipo de archivos y separar sus atributos es trivial.

```
1 f = open('./estudiantes.txt')
2 for l in f:
3   n = l.split(';')
4   print("Nombre:",n[0])
5   print("Apellido:",n[1])
6   print("N alumno:",n[2])
7   print("Notas:",n[3],n[4],n[5])
8 f.close()
```

Ejemplo

Usemos lo aprendido para calcular las notas finales a partir del archivo `"./notas.txt"`.

Ejemplo

Usemos lo aprendido para calcular las notas finales a partir del archivo `"./notas.txt"`.

Formato: `"num_alumno C1 C2 C3 MT Ex Bn"`.

```
14620405 1 6.6 4.7 5.4 5.2 1
14622106 1 3.9 3.4 6.5 6.4 1
14620162 1 6.2 6.7 5.8 4.9 0
14623250 2 4.1 5.0 3.6 4.7 1
14621258 2 5.5 6.9 3.5 4.0 0
14622823 1 4.7 6.9 4.5 6.6 0
14623102 4 4.5 4.8 5.9 3.3 1
```

`./notas.txt`

Ejemplo

```
1  def calcular_nota_final(n):
2    Bn = int(n[6]); MT = float(n[4]); Ex = float(n[5])
3    C = sum([float(n[1]),float(n[2]),float(n[3])])/3
4    C += int(Bn)*0.5
5    nota_final = min([0.3*C + 0.3*MT + 0.4*Ex,7])
6    if(C<4 or Ex<4): nota_final=min([nota_final,3.9])
7    return nota_final
8
9  f = open("./notas.txt") # Abrir archivo
10 suma = 0; total = 0; reprobados = 0
11 for l in f:                  # Recorro líneas
12   n = l.strip().split(' ')
13   nf = calcular_nota_final(n)
14   if(nf < 4): reprobados+=1
15   suma += nf; total += 1
16   print(n[0],"%0.1f"%nf)
17 f.close() # Cerrar archivo
18 print("promedio","%0.2f"%(suma/total))
19 print("reprobados",reprobados)
```

Escritura de archivos

Escritura de archivos

`f = open(p,m)`: Función que abre un archivos en modo `m`.

- `"r"` → Modo lectura (por defecto).
- `"w"` → Modo escritura.
- `"a"` → Modo append.

Escritura de archivos

`f = open(p,m)`: Función que abre un archivos en modo `m`.

- `"r"` → Modo lectura (por defecto).
- `"w"` → Modo escritura.
- `"a"` → Modo append.

`f.write(s)`: Escribe `s` en el archivo.

`f.close()`: Método que cierra el archivo.

Escritura de archivos

¿Diferencia entre modo *escritura* y modo *append*?

Escritura de archivos

¿Diferencia entre modo *escritura* y modo *append*?

`f = open(p,"w")`: Crea un nuevo archivo en ruta `p`. Si el archivo ya existe, borra su contenido.

Escritura de archivos

¿Diferencia entre modo *escritura* y modo *append*?

`f = open(p,"w")`: Crea un nuevo archivo en ruta `p`. Si el archivo ya existe, borra su contenido.

`f = open(p,"a")`: Crea un nuevo archivo en ruta `p`. Si el archivo ya existe, lo lee y agrega el nuevo contenido al final.

Escritura de archivos

Ejemplo: Escribir archivo con números del 1 al 5.

Escritura de archivos

Ejemplo: Escribir archivo con números del 1 al 5.

```
1 f = open('./test.txt','w')
2 for i in range(1,6):
3   f.write(str(i) + "\n")
4 f.close()
```

Escritura de archivos

Ejemplo: Escribir archivo con números del 1 al 5.

```
1 f = open('./test.txt','w')
2 for i in range(1,6):
3   f.write(str(i) + "\n")
4 f.close()
```

Obs: A diferencia del `print()`, `write()` no hace un salto de línea automático.

Escritura de archivos

Resultado: (Luego de ejecutarlo 2 veces)

```
1 1
2 2
3 3
4 4
5 5
```

`"./test.txt"`

Escritura de archivos

¿Qué ocurre si hago lo mismo, pero en modo *append*?

```
1 f = open('./test2.txt','a')
2 for i in range(1,6):
3   f.writel(str(i) + "\n")
4 f.close()
```

Escritura de archivos

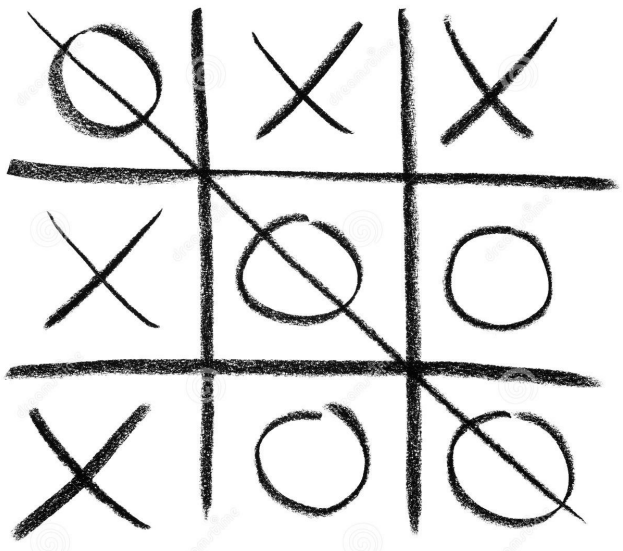
Resultado: (Luego de ejecutarlo 2 veces)

```
1 1
2 2
3 3
4 4
5 5
6 1
7 2
8 3
9 4
10 5
```

"./test2.txt"

Ejemplo: Gato

Guardar la partida del gato.



Ejemplo: Gato

¿Cómo funciona el gato?

Ejemplo: Gato

¿Cómo funciona el gato?

- Tablero es una lista con números del 1 al 9.
- En variable `turno` almaceno el turno actual (`"x"` o `"o"`).
- En cada turno cambio un número del tablero por `turno`.
- Verifico si alguien gana o es empate.
- Cambio de turno.

Motivación

```
38 turno = x; tablero = []
39 for i in range(1,10):
40   tablero.append(str(i))
41
42 while(True):
43   mostrar_tablero(tablero)
44   # Realizo movimiento
45   print("Turno",turno)
46   pos = int(input("Ingrese movida:"))
47   if(tablero[pos-1] == str(pos)):
48     tablero[pos-1] = turno
49   # Veo si se acabó el juego o es empate
50   if(hay_ganador(tablero)):
51     print("Gato! ganó", turno); break
52   if(es_empate(tablero)):
53     print("Empate!"); break
54   # Cambio de turno
55   if(turno == x): turno = o
56   else: turno = x
```

Motivación

¿Cómo guardamos el estado del gato?

Motivación

¿Cómo guardamos el estado del gato?

- Recordar quién tiene el turno.
- Recordar el tablero actual.
- Olvidar todo cuando el juego finalice.

Ejemplo: Gato

Idea:

- Guardar archivo `juego.txt` con el tablero actual y el turno.
- Leer archivo `juego.txt` para retomar juego anterior.

Ejemplo: Gato

Idea:

- Guardar archivo `juego.txt` con el tablero actual y el turno.
- Leer archivo `juego.txt` para retomar juego anterior.

Problemas:

- ¿Qué pasa si no existe `juego.txt`?
- ¿Qué pasa si alguien gana el gato?

Ejemplo: Gato

Idea:

- Guardar archivo `juego.txt` con el tablero actual y el turno.
- Leer archivo `juego.txt` para retomar juego anterior.

Problemas:

- ¿Qué pasa si no existe `juego.txt`?
- ¿Qué pasa si alguien gana el gato?

Funciones útiles:

- `os.path.exists(p)`: Retorna `True` si `p` existe.
- `os.remove(p)`: Borra el archivo `p`.

Ejemplo: Gato

Dejo en `backup` la ruta al archivo de respaldo.

```
52 backup = './juego.txt'
```

Ejemplo: Gato

Dejo en `backup` la ruta al archivo de respaldo.

```
52 backup = './juego.txt'
```

Guardar el tablero actual (luego de cada cambio de turno):

```
46 def guardar(path, tablero,turno):
47   f = open(path,'w')
48   f.write(turno + ";" + ";".join(tablero))
49   f.close()
```

Ejemplo: Gato

Dejo en `backup` la ruta al archivo de respaldo.

```
52 backup = './juego.txt'
```

Guardar el tablero actual (luego de cada cambio de turno):

```
46 def guardar(path, tablero,turno):
47   f = open(path,'w')
48   f.write(turno + ";" + ";".join(tablero))
49   f.close()
```

Borrar archivo al finalizar un juego:

```
99 os.remove(backup)
```

Ejemplo: Gato

Cargar tablero antiguo o crear un tablero nuevo.

```
31 def cargar(path, x, o):
32   # Creo tablero nuevo
33   turno = x
34   tablero = []
35   for i in range(1,10):
36     tablero.append(str(i))
37   # si hay tablero guardado lo cargo
38   if(os.path.exists(path)):
39     # sino, creo un nuevo tablero
40     f = open(path)
41     datos = f.readline().strip().split(';')
42     f.close()
43     turno,tablero = datos[0], datos[1:]
44   return turno,tablero
```

Ejemplo: Gato

Cargar tablero antiguo o crear un tablero nuevo.

```
31 def cargar(path, x, o):
32   # Creo tablero nuevo
33   turno = x
34   tablero = []
35   for i in range(1,10):
36     tablero.append(str(i))
37   # si hay tablero guardado lo cargo
38   if(os.path.exists(path)):
39     # sino, creo un nuevo tablero
40     f = open(path)
41     datos = f.readline().strip().split(';')
42     f.close()
43     turno,tablero = datos[0], datos[1:]
44   return turno,tablero
```

Código completo disponible en el siding...

Ejercicios

1) Lea el archivo `Einstein.txt` y muestre en consola sólo las frases con menos de 50 caracteres.

2) Cree un programa que reciba la ruta a un archivo y muestre el número de líneas que posee.

3) Agregue al gato un historial de victorias, empates y derrotas (perdurable en el tiempo).

4) Cree un programa que permita moverse por la jerarquía de carpetas del sistema de archivos. Le debe ofrecer al usuario 4 opciones: Listar elementos de la carpeta actual, entrar a una sub-carpeta, volver a la carpeta padre y mostrar el contenido de un archivo.