Los programas de Computador

Jhon J. Padilla A., PhD.

El lenguaje Python

- En nuestro curso usaremos el lenguaje Python, por ser un lenguaje sencillo y muy usado en el mundo actual.

Aplicaciones desarrolladas en Python

BitTorrent™

Calibre ebook reader

django

Ingresar al intérprete de Python

- La instalación del intérprete de Python no se explica aquí.
- En algún momento, terminarás en un terminal o ventana de comandos, escribirás '*python*', y el intérprete de Python comenzará a ejecutarse en modo interactivo, apareciendo algo como lo siguiente:

```
Python 2.6.1 (r261:67515, Jun 24 2010, 21:47:49)
[GCC 4.2.1 (Apple Inc. build 5646)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Interacción con el intérprete de Python

- El prompt o indicador >>> es el modo que tiene el intérprete de Python de preguntarte: “¿Qué quieres que haga a continuación?”. Python está preparado para tener una conversación contigo. Todo lo que tienes que hacer es hablar el idioma de Python.

Nuestro primer intento...

- Imaginemos por ejemplo que no conoces ni siquiera la más simple de las palabras o frases del lenguaje Python. Tal vez quieras usar la línea habitual que siguen los astronautas cuando aterrizan en un planeta remoto y quieren hablar con sus habitantes:

```
>>> Venimos en son de paz, por favor llevadnos ante vuestro lider
  File "<stdin>", line 1
    Venimos en son de paz, por favor llevadnos ante vuestro lider
         ^
SyntaxError: invalid syntax
>>>
```

- Esto no está funcionando!!!, Python no entendió nada....

Ahora intenta....

```
>>>print ('Hola Mundo')

Hola Mundo

>>>
```

- Parece que ahora si funcionó!...el intérprete de Python ha entendido que quieres que imprima en pantalla la frase “Hola, mundo!”

Ahora intenta....

```
>>> print('Vamos a tener un festín esta noche a menos que nos digas)
  File "<stdin>", line 1
    print 'Vamos a tener un festín esta noche a menos que nos digas
                                                                   ^
SyntaxError: EOL while scanning string literal
>>>
```

Ahora intenta....

```
>>> print('Vamos a tener un festín esta noche a menos que nos digas)
  File "<stdin>", line 1
    print 'Vamos a tener un festín esta noche a menos que nos digas
                                                                   ^
SyntaxError: EOL while scanning string literal
>>>
```

- Parece que hubo un error...faltó cerrar las comillas de la frase y Python no pudo entender qué querías decir...
- En este momento, ya deberías haberte dado cuenta de que, a pesar de que Python es increíblemente complejo, potente y muy exigente con la sintaxis que debes usar para comunicarte con él, Python no es inteligente.

Cómo salir del intérprete de Python

- Antes de terminar nuestra primera conversación con el intérprete de Python, probablemente debas saber cual es el modo correcto de decir “adios” cuando estás interactuando con los habitantes del Planeta Python:

```
>>> adios
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'adios' is not defined

>>> if you don't mind, I need to leave
  File "<stdin>", line 1
    if you don't mind, I need to leave
             ^
SyntaxError: invalid syntax

>>> quit()
```

Cómo salir del intérprete de Python

- Te habrás dado cuenta de que el error es diferente en los primeros dos intentos, a pesar de ser ambos incorrectos.
- El segundo error es diferente porque **if** es una palabra reservada, y Python vió la palabra reservada en la frase y creyó que estabas intentando decirle algo, pero encontró la sintaxis de la sentencia incorrecta.
- El modo correcto de decir “adios” a Python es introducir **quit()** en el indicador interactivo >>>.
- Probablemente te hubiera llevado un buen rato adivinarlo, así que

  es posible que el tener un libro a mano esté empezando a resultarte útil.

Variables, Expresiones y Sentencias en Python

Variables

- La información se almacena en la memoria del computador en forma de variables
- Una variable tiene un nombre simbólico:
  - A=5
  - opcion=”Y”
- Una variable es una parte de la memoria que almacena algún tipo de información
- Hay diferentes tipos de información o variables

Ejemplos de contenidos de las variables

- Cadenas de caracteres:
  - “Hola mundo”
- Numéricas:
  - Enteros (Integer): 432
  - Reales (float): -12,425781
- Booleanas:
  - Sólo toman dos posibles valores: Falso o Verdadero

Caracteres ASCII y cadenas de caracteres

Caracter

- Caracter: símbolo usado en escritura humana:
  - A, b, d, =, /
- También es un símbolo que tiene significado dentro de un archivo de texto (ej: nueva línea, enter, fin de archivo)
- Cada carácter se almacena en forma de un conjunto de unos y ceros en la memoria de un computador, el significado es diferente que el código binario.
- Un computador puede traducir los unos y ceros a la forma de cada carácter para imprimirlo en pantalla o en una impresora.
- También se puede almacenar en la memoria el código del caracter de cada tecla del teclado de un computador.

Código ASCII

- Es un estándar que indica cuáles son las combinaciones de unos y ceros a utilizar para cada carácter.
- Lo definió la ANSI (American National Standards Institute)
- ASCII: American Standard Code for Information Interchange
- Se utiliza en todo el mundo para almacenar texto en los archivos o en la memoria del computador.

Tabla de Caracteres ASCII

El código ASCII

sigla en inglés de American Standard Code for Information Interchange
(Código Estadounidense Estándar para el Intercambio de Información)

Tomada del sitio:
tecnologiaeinformaticaa.es.tl

Caracteres ASCII de control

00	NULL	(carácter nulo)
01	SOH	(inicio encabezado)
02	STX	(inicio texto)
03	ETX	(fin de texto)
04	EOT	(fin transmisión)
05	ENQ	(consulta)
06	ACK	(reconocimiento)
07	BEL	(timbre)
08	BS	(retroceso)
09	HT	(tab horizontal)
10	LF	(nueva línea)
11	VT	(tab vertical)
12	FF	(nueva página)
13	CR	(retorno de carro)
14	SO	(desplaza afuera)
15	SI	(desplaza adentro)
16	DLE	(esc.vínculo datos)
17	DC1	(control disp. 1)
18	DC2	(control disp. 2)
19	DC3	(control disp. 3)
20	DC4	(control disp. 4)
21	NAK	(conf. negativa)
22	SYN	(inactividad sínc)
23	ETB	(fin bloque trans)
24	CAN	(cancelar)
25	EM	(fin del medio)
26	SUB	(sustitución)
27	ESC	(escape)
28	FS	(sep. archivos)
29	GS	(sep. grupos)
30	RS	(sep. registros)
31	US	(sep. unidades)
127	DEL	(suprimir)

Caracteres ASCII imprimibles

32	espacio	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

ASCII extendido (Página de código 437)

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

Cadenas de Caracteres

- Consisten de una secuencia de caracteres ASCII
- Se almacenan en variables que se denominan cadenas de caracteres
- Los caracteres de la cadena deben ir entre comillas para indicar al intérprete dónde inicia y dónde termina la cadena
- Ej: 'Hola Mundo'

Imprimir variables y determinar su tipo

```
python
>>> print(4)
4
```

Inicia o ejecuta Python

Imprime el número 4

Para saber qué tipo de variable es una variable particular:

```
>>> type('¡Hola, mundo!')
<type 'str'>
>>> type(17)
<type 'int'>

>>> type(3.2)
<type 'float'>
```

???

???

???

Imprimir variables y determinar su tipo

Para saber qué tipo de variable es una variable particular:

```
>>> type('¡Hola, mundo!')
<type 'str'>
>>> type(17)
<type 'int'>

>>> type(3.2)
<type 'float'>
```

Cadena de caracteres

Tipo Entero

Tipo Float

Imprimir variables y determinar su tipo

Ahora teclea lo siguiente:

```
>>> type('17')
<type 'str'>
>>> type('3.2')
<type 'str'>
```

No eran números?.....por qué dice que es str?

Imprimir variables y determinar su tipo

Ahora teclea lo siguiente:

```
>>> type('17')
<type 'str'>
>>> type('3.2')
<type 'str'>
```

No eran números?.....por qué dice que es str?

Respuesta:

Los valores están entre comillas simples, por lo que Python interpreta que son cadenas de caracteres!!!!

Sentencias de Asignación

Una sentencia de asignación crea variables nuevas y les da valores.

Escribe lo siguiente en el intérprete de python:

```
>>> mensaje = 'Y ahora algo completamente diferente'
>>> n = 17
>>> pi = 3.1415926535897931
```

Ahora veremos el valor de las variables creadas y sus valores asignados:

```
>>> print(n)
17
>>> print(pi)
3.14159265359
```

Practicando con los nombres de variables

Ahora digita estas líneas....¿Por qué te sale error de sintaxis?

```
>>> 76trombones = 'gran desfile'
SyntaxError: invalid syntax
>>> more@ = 1000000
SyntaxError: invalid syntax
>>> class = 'Teorema avanzado de Zymurgy'
SyntaxError: invalid syntax
```

Practicando con los nombres de variables

Ahora digita estas líneas....¿Por qué te sale error de sintaxis?

```
>>> 76trombones = 'gran desfile'
SyntaxError: invalid syntax
>>> more@ = 1000000
SyntaxError: invalid syntax
>>> class = 'Teorema avanzado de Zymurgy'
SyntaxError: invalid syntax
```

Es incorrecto porque comienza con un número

Es incorrecto porque contiene un carácter no permitido: @

Es incorrecto porque es una palabra reservada de Python

Palabras reservadas de Python

```
and       del       from      not       while
as        elif      global    or        with
assert    else      if        pass      yield
break     except    import    print
class     exec      in        raise
continue  finally   is        return
def       for       lambda    try
```

Nombres de las variables en Python

- Los programadores generalmente eligen nombres para sus variables que tengan sentido y documenten para qué se usa esa variable.
- Los nombres de las variables pueden ser arbitrariamente largos.
- Pueden contener tanto letras como números, pero no pueden comenzar con un número.
- Se pueden usar letras mayúsculas, pero es buena idea comenzar los nombres de las variables con una letras minúscula (veremos por qué más adelante).
- El caracter guión-bajo (_) puede utilizarse en un nombre. A menudo se utiliza en nombres con múltiples palabras, como en *mi_nombre o velocidad_de_golondrina_sin_carga*.
- Los nombres de las variables pueden comenzar con un caracter guión-bajo, pero generalmente se evita usarlo así a menos que se esté escribiendo código para librerías que luego utilizarán otros.

Los programas

Qué es un programa?

- Escribir frases en el intérprete de Python es una buena forma de experimentar con las características de Python, pero no resulta recomendable para resolver problemas de cierta complejidad.
- Por lo general necesitamos que el computador lea y ejecute automáticamente muchas instrucciones con el fin de que lo haga rápido.
- Si damos instrucción por instrucción a la velocidad que digitamos, no estaríamos aprovechando la rapidez del computador.

Qué es un programa?

- Un programa es un listado de instrucciones escritas una tras otra en un archivo de texto.
- El computador puede ir leyendo y ejecutando las instrucciones una por una a una velocidad increíble.
- El archivo de texto que contiene las instrucciones en Python recibe el nombre de *script*.
- Se recomienda ponerle un nombre al archivo con extensión “.py” para indicar que está escrito en Python. (ej: “archvo.py”)

Ejemplo de un programa en Python

- Este programa cuenta las palabras en un archivo de texto e imprime la palabra que más se repite
- (Por ahora no lo explicaremos)
- Sólo escribe las instrucciones en un editor de texto plano y guárdalo en un nombre como *words.py* y luego ejecuta en el terminal el comando: *python words.py*

```
nombre =str(input('Introduzca fichero:'))
manejador = open(nombre, 'r')
texto = manejador.read()
palabras = texto.split()
contadores = dict()
for palabra in palabras:
   contadores[palabra] = contadores.get(palabra,0) + 1
mayorcantidad = None
mayorpalabra = None
for palabra,contador in contadores.items():
    if mayorcantidad is None or contador > mayorcantidad:
        mayorpalabra = palabra
        mayorcantidad = contador
print (mayorpalabra, mayorcantidad)
```

Cómo desarrollar los programas?

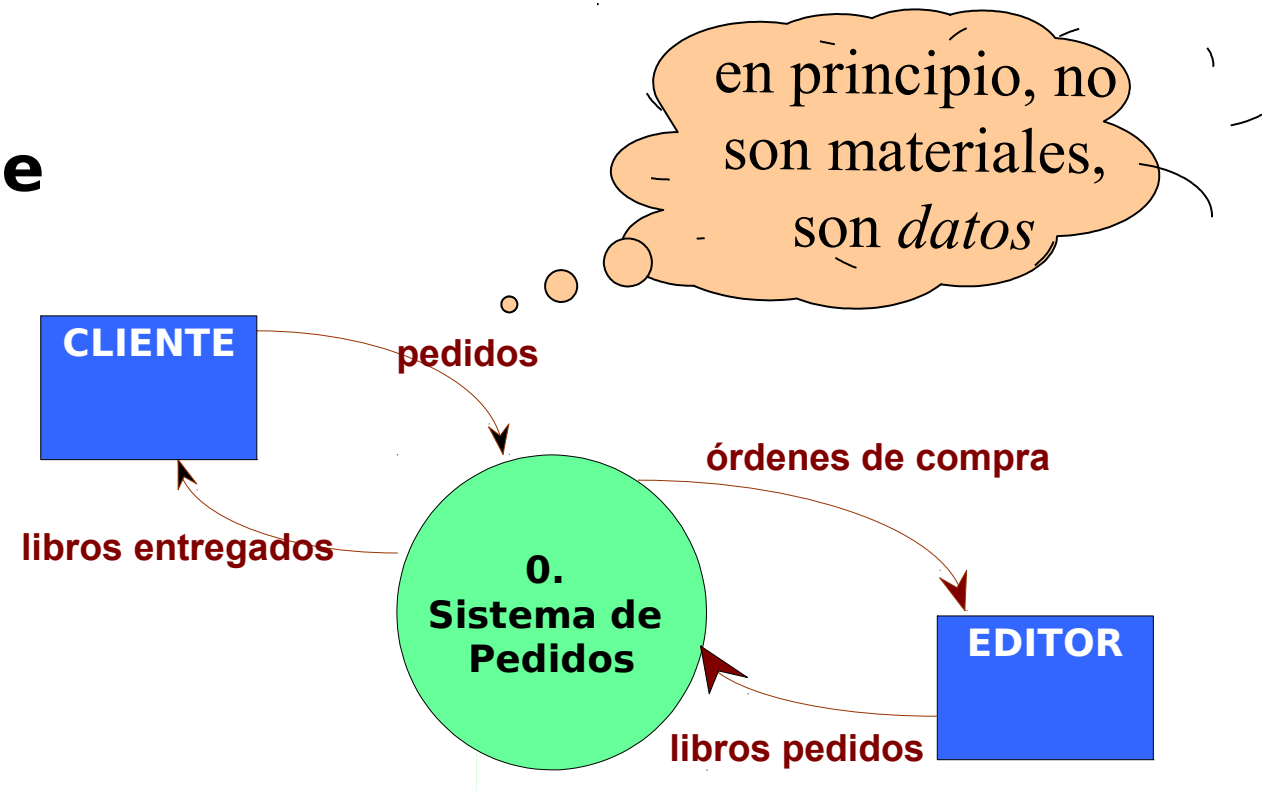
Ciclo de vida del software

Análisis de requerimientos

- Definir el problema y las características de la solución no es tarea fácil.
- El problema a resolver se define junto con el cliente.
- Se deben detallar los diferentes requisitos que el cliente exige para la solución.
- Se usan herramientas de modelado (Diagramas, descripciones textuales) para definir claramente qué quiere el cliente.
- Se podría generar una interfaz de usuario vacía para generar un prototipo de cómo debería quedar el SW en apariencia.

Ejemplo de el análisis de requisitos

Diagrama de contexto

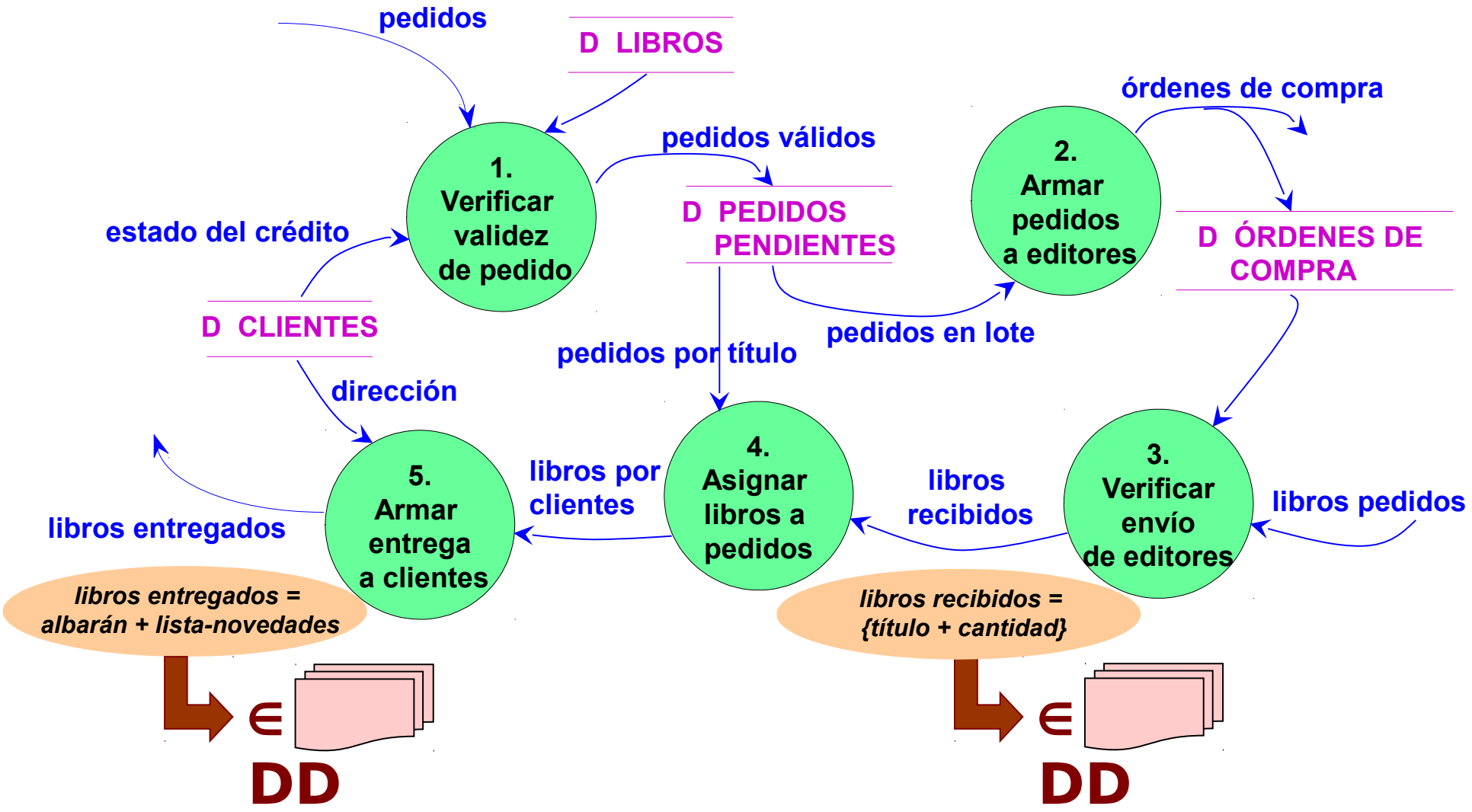


Diseño del SW

- Una vez definidas las características del problema y la solución, se pasa a definir cómo estará conformada la solución internamente y cómo debe operar para que dé la respuesta deseada por el cliente.
- Estrategia: Divide y vencerás.
- Como salida de esta etapa se definen varios módulos internos del software y cómo interactúan entre ellos.
- Para cada módulo se describe lo que hace mediante un algoritmo (pseudocódigo o Diagrama de flujo)

Ejemplo del Diseño

1. Sistema de pedidos



Ejemplo de un algoritmo para encontrar la silla en un cine

1. **inicio** //algoritmo para encontrar la butaca del espectador
2. caminar hasta llegar a la primera fila de butacas
3. **repetir**
   compara número de fila con número impreso en billete
   si no son iguales, **entonces** pasar a la siguiente fila
   hasta_que se localice la fila correcta
4. **mientras** número de butaca no coincida con número de billete
   hacer avanzar a través de la fila a la siguiente butaca
   fin-mientras
5. sentarse en la butaca
6. **fin**

Codificación

- Para el algoritmo de cada módulo definido en la etapa de diseño, deberá hacerse un programa descrito mediante las instrucciones propias de un lenguaje de alto nivel.
- Lenguajes de alto nivel:
  - C
  - C++
  - Java
  - Perl
  - Python
  - Visual Basic
  - Etc.

Ejemplos codificación en lenguajes Pascal y C

Cuadro Comparativo de Instrucciones fundamentales en Pascal y C

Instrucciones	Pascal	C
Entrada	Read (a);	Scanf("%d", &a);
Salida	Write (a);	Printf ("%d", a);
Atribución	b := a;	b = a;

Transferencia Condicional	If condición
	Then
	begin
	Proceso1;
	end
	Else
	begin
	Proceso2;
	end;
		If (condición)
		{
		Proceso1;
		}
		else
		{
		Proceso2;
		{

Repetición	for i :=1 to n do
	begin
	Proceso;
	end;
		for (i = 0; i < n; i++)
		{
		Proceso;
		}

	While condición do
	Begin
	Proceso;
	End;
		While (condición)
		{
		Proceso;
		}

	Repeat Proceso; Until condición;	Do { Proceso; } While (condición);

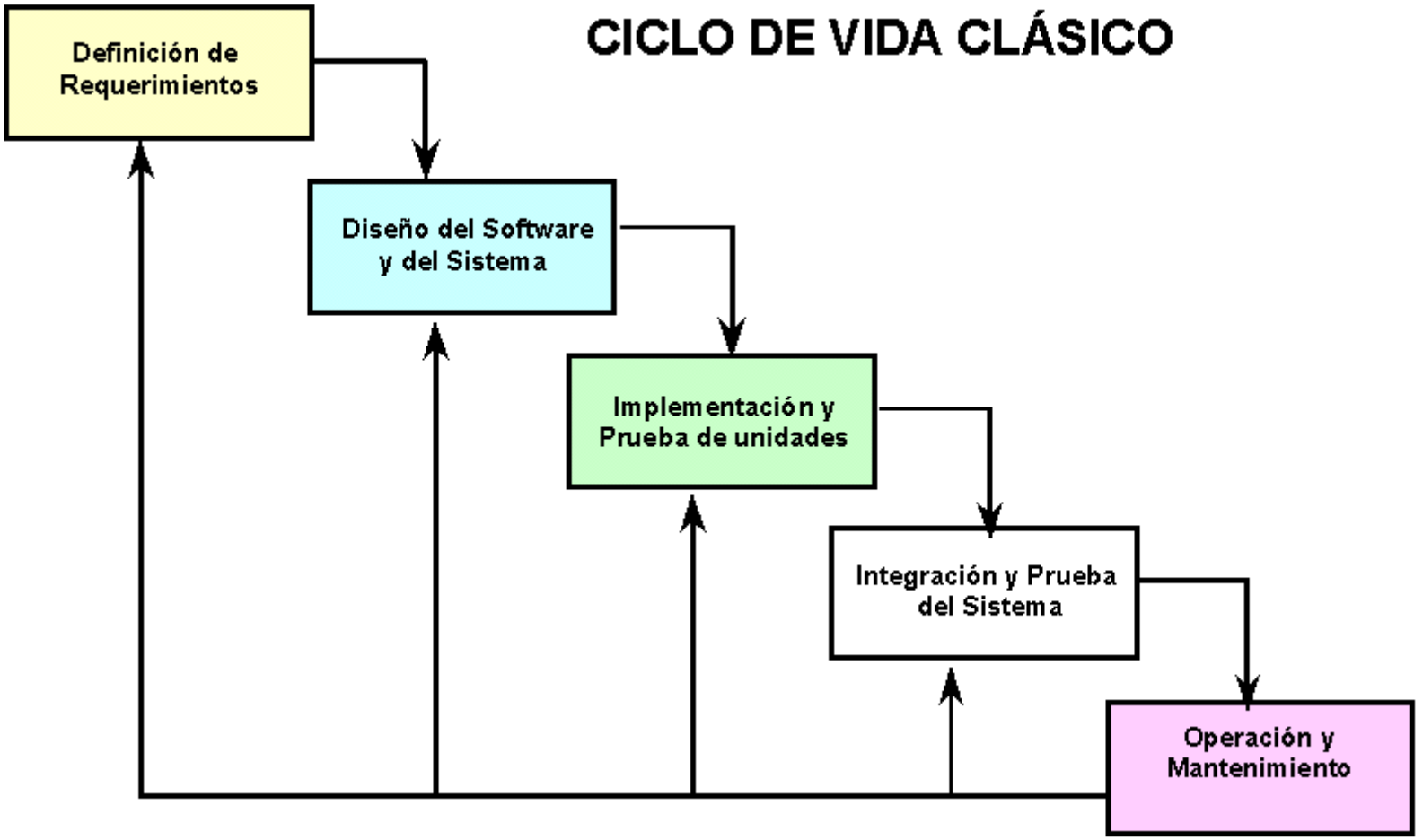
Pruebas (Depuración)

- Las pruebas consisten en la ejecución del programa y la verificación de que esté realizando la tarea correctamente.
- De haber errores, deberán hacerse correcciones y volver a ejecutar el programa.
- Este proceso se realiza tantas veces como sea necesario hasta obtener una ejecución perfecta del programa.

Mantenimiento

- Consiste en hacer modificaciones al programa en la medida que se requiera hacer cambios en su comportamiento por requerimiento de los clientes.
- Esto se hace durante la vida útil del programa, que puede ser por años.

Ciclo de vida del software



Ciclo de vida del software