



SISTEMAS DE GESTIÓN DE BASES DE DATOS AVANZADOS

V2022



JAVIER ROA BENITEZ

Licenciado en Análisis de Sistemas

Coach Ontológico Organizacional ***Certificado FICOP 2848***

Conferencista y Capacitador

Nacido en Villarrica del Espíritu Santo el 21/09/1965

Autor del libro: "Preguntas que pueden mejorar tu vida"

ESPECIALIZACIONES:

- Didáctica Superior Universitaria
- Población y Desarrollo
- Investigación Socio Demográfica

TRABAJOS ACTUALES:

- **Consultor Informático, con especialización en:** Telecomunicaciones, Gobierno Electrónico, Compras Públicas y Nuevas Tecnologías de Desarrollo de Software, Sistemas para COOPERATIVAS

CONSULTORÍAS REALIZADAS:

- **A Nivel Nacional:**
 - **Proyectos Financiados por el BID y Banco Mundial, USAID :** Ministerio de Hacienda, Contrataciones Publicas, SICIAP(del MSPBS)
- **A nivel Internacional:**
 - **Telecel Bolivia (Bolivia)** - Desarrollo de Software de Telefonía
 - **Trilogy Partners (Rca. Dominicana)** - Desarrollo de Software de Telefonía
 - **Finacus (India)** - Trabajos de diseño, desarrollo y tropicalización de Productos Financieros/Digitales

JAVIER ROA BENITEZ

Soy un ser FELIZ, satisfecho con la vida, me siento exitoso con lo que ya tengo, pero aún tengo sueños y voy por ellos , sin que estos me quiten el sueño.

Soy apasionado por mis profesiones, comprometido con cada una de las tareas y actividades que me toca hacer, comprendiendo que cada trabajo que realizo tiene un impacto en los seres humanos más allá de las tareas que realizo.

Tengo más de 30 años de experiencia en el área informática, analizando, diseñando, desarrollando e implementando soluciones para las empresas, en el 2017 descubrí mi verdadero propósito que lo desarrollo por medio del Coaching Ontológico, acompañando a las personas y organizaciones en la búsqueda de resultados extraordinarios, mediante conversaciones poderosas, talleres, cursos y conferencias.

OBJETIVOS

Al final del curso el alumno será capaz de:

- Comprender las características y potencialidades de un lenguaje orientado a objetos
- Analizar y resolver problemas para desarrollar un programa en Python
- Desarrollar programas en Python que conectan con bases de datos en Postgresql

COMPETENCIAS

Competencias específicas

- Resolver problemas y desarrollarlos con el lenguaje Python
- Conectar sus programas con base de datos PostgreSQL
- Construir sistemas basados en Python/Postgresql

Competencias transversales

- Utilizar tecnologías de la información y de la comunicación.
- Desarrollar la habilidad para trabajar en equipos multidisciplinarios.
- Actuar de conformidad a las normas éticas establecidas.
- Demostrar razonamiento crítico y objetivo.
- Identificar, plantear y resolver problemas.
- Desarrollar habilidades de escucha activa
- Comunicar sus ideas, proyectos e inquietudes de manera asertiva

JIT-CITA 2019

Conferencias

- Impacto de las tecnologías para la comunicación y el acceso en la calidad de vida de niños con discapacidad,
- Model Driven Engineering (and Web Engineering). Promise or reality? Y
- Testing Ágil e integración continua

Impacto de las tecnologías para la comunicación y el acceso en la calidad de vida de niños con discapacidad,

- La herramienta **IDH**(Índice de desarrollo humano) utiliza variables de cumplimiento básicamente en SALUD, EDUCACION y NIVEL DE VIDA
- **World Happiness Report** (Informe sobre la felicidad en el mundo), donde Paraguay figura en el puesto 63 de 156, usa como variables para definir las posiciones: Dinero que ganan, soporte social, salud, libertad, generosidad y ausencia de corrupción en su país
- **Aquí se mide mas el BIENESTAR SUBJETIVO que la FELICIDAD** pues la felicidad lo construye cada ser humano y no depende de nadie más.

En nuestro deseo de llevar a niveles científicos todo, nos olvidamos que existen características del ser humano que solo pueden ser percibidas, más que medidas

En cuanto a la necesidad de “humanizar” a los técnicos:

- En la conferencia MDE(Model Driven Engineering)
 - Se mostro los motivos por los cuales los informáticos no cumplimos con los tiempos comprometidos, en el me llamo la atención que los errores de código solo impactan en un 7% , en cambio los errores de requerimiento afectan en un 56%, en otro grafico que se mostraba también se ve en una línea de tiempo como los requerimientos van cambiando y creciendo en el tiempo, esto nos demuestra que no estamos entendiendo bien los requerimientos en las primeras etapas
 - Para entender un requerimiento lo que necesitamos es SABER ESCUCHAR y aplicar técnicas de escucha activa con el cliente
 - Muchos profesionales estamos muy bien preparados técnicamente, en cambio no sabemos escuchar, entonces entrevistamos a nuestros clientes desde a posición de SABELOTODOS, nos piden algo y ya sabemos la solución, inclusive se lo contamos, es así que el cliente se calla y ya no da mas detalles (para que si esta frente a un experto) y eso hace que nos perdemos aspectos importantes de lo que realmente desea

En cuanto a la necesidad de “humanizar” a los técnicos:

- **Testing Ágil e integración continua**
 - Se habló también de la filosofía DEVOPS (que incluye seguridad, maneras colaborativas de trabajo, análisis de datos y muchas otras cosas.).
 - En esta filosofía además de las cuestiones técnicas, se habla de la necesidad de un cambio cultural, un involucramiento de equipo, esto es:

Nivel Organizacional:

- *No tener silos,*
- *confianza en los equipos,*
- *auto organización y autonomía*

Nivel de Equipos:

- *Compartir las responsabilidades (desarrollo/operación y QA)*
- *Responsabilidad más allá de la entrega,*
- *Compromiso y no solo cumplimiento (donde cumplo y miento).*



Javier Roa Benítez

jroabenitez@gmail.com

Cel: +595981407450

Contenido:

1. Abriendo tu apetito
2. Usando el intérprete de Python
3. Una introducción informal a Python
4. Herramientas para control de flujo
5. **Estructuras de datos**
6. Módulos
7. Entrada y salida
8. Errores y excepciones
9. Clases
10. Conexión con Bases de Datos
11. Introducción a Django(Framework)

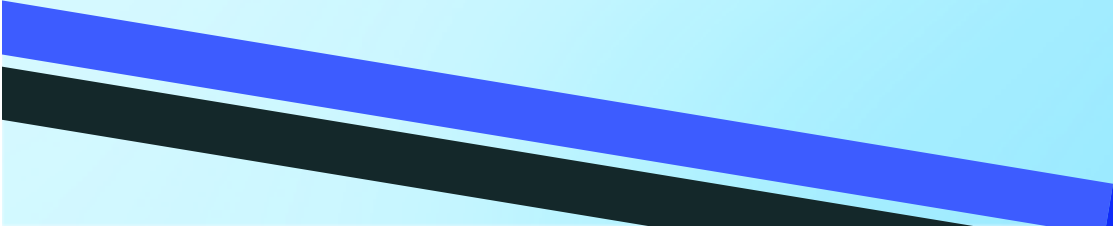
Muuuchos ejercicios.....

Reglas de Convivencia

- Asistencia
- Celulares
- Uso de las maquinas
- Desarrollo de clases
- Exámenes
- Preguntas???

Abriendo el apetito

- CUANDO TRABAJAS MUCHO EN LA COMPUTADORA, te gustaría automatizar alguna tarea en la computadora?:
 - realizar una búsqueda y reemplazo en un gran número de archivos de texto
 - renombrar y reorganizar un montón de archivos con fotos de una manera compleja
 - escribir alguna pequeña base de datos personalizada
 - una aplicación especializada con interfaz gráfica
 - un juego simple.
- Si sos un desarrollador de software profesional Y:
 - Necesitas trabajar con varias bibliotecas de C/C++/Java pero encuentres que se hace lento el ciclo usual de escribir/compilar/testear/recompilar.
 - estás escribiendo una batería de pruebas para una de esas bibliotecas y encuentres que escribir el código de testeo se hace una tarea tediosa.
 - has escrito un programa al que le vendría bien un lenguaje de extensión, y no quieres diseñar/implementar todo un nuevo lenguaje para tu aplicación.



Python

Es para vos.....

Que puedes hacer con Python?

- Podrías escribir un script (o programa) en el interprete de comandos o un archivo por lotes de Windows para algunas de estas tareas, pero los scripts se lucen para mover archivos de un lado a otro y para modificar datos de texto, no para aplicaciones con interfaz de usuario o juegos.
- Podrías escribir un programa en C/C++/Java, pero puede tomar mucho tiempo de desarrollo obtener al menos un primer borrador del programa.
- Python es más fácil de usar, está disponible para sistemas operativos Windows, Mac OS X y Unix, y te ayudará a realizar tu tarea más velozmente.

Como es Python?

- Python es fácil de usar, pero es un lenguaje de programación de **verdad**, ofreciendo mucho mayor estructura y soporte para programas grandes que lo que lo que pueden ofrecer los scripts de Unix o archivos por lotes.
- Python ofrece mucho más chequeo de error que C, y siendo un *lenguaje de muy alto nivel*, tiene tipos de datos de alto nivel incorporados como arreglos de tamaño flexible y diccionarios.
- Debido a sus tipos de datos más generales Python puede aplicarse a un dominio de problemas mayor que Awk o incluso Perl, y aún así muchas cosas siguen siendo al menos igual de fácil en Python que en esos lenguajes

Como es Python?

- Te permite separar tu programa en módulos que pueden reusarse en otros programas en Python.
- Viene con una gran colección de módulos estándar que puedes usar como base de tus programas, o como ejemplos para empezar a aprender a programar en Python.

Algunos de estos módulos proveen cosas como

- entrada/salida a archivos,
- llamadas al sistema,
- sockets,
- incluso interfaces a sistemas de interfaz gráfica de usuario como Tk.

Como es Python?

- *Python es un lenguaje interpretado*, lo cual puede ahorrarte mucho tiempo durante el desarrollo ya que no es necesario compilar ni enlazar.
- El intérprete puede usarse interactivamente, lo que facilita experimentar con características del lenguaje, escribir programas descartables, o probar funciones cuando se hace desarrollo de programas de abajo hacia arriba.
- Es también una calculadora de escritorio práctica.
- Permite escribir programas compactos y legibles

Como es Python?

- Los programas en Python son típicamente más cortos que sus programas equivalentes en C, C++ o Java por varios motivos:
 - los tipos de datos de alto nivel permiten expresar operaciones complejas en una sola instrucción
 - la agrupación de instrucciones se hace por sangría en vez de llaves de apertura y cierre
 - no es necesario declarar variables ni argumentos.

Python, el nombre de una Serpiente?

El lenguaje recibe su nombre del programa de televisión de la BBC “Monty Python’s Flying Circus” y no tiene nada que ver con reptiles.

Hacer referencias a sketches de Monty Python en la documentación no sólo esta permitido, ¡sino que también está bien visto!

Monty Python's Flying Circus es una serie de televisión británica creada y protagonizada por el grupo de humoristas Monty Python y consta de 4 temporadas, con un total de 45 capítulos. Esta se basaba en sketches breves que en muchas ocasiones incluían una importante carga de crítica social, y la mayoría de las obras rozaban el absurdo total

Usando el intérprete de Python

Invocando al Interprete

- **En máquinas con LINUX:** Por lo general, el intérprete de Python se instala en `file:/usr/local/bin/python` en las máquinas dónde está disponible; poner `/usr/local/bin` en el camino de búsqueda de tu intérprete de comandos Unix hace posible iniciarlo ingresando la orden:

`python`

- **En máquinas con Windows,** la instalación de Python por lo general se encuentra en `C:\Python27`, aunque se puede cambiar durante la instalación. Para añadir este directorio al camino, puedes ingresar la siguiente orden en el prompt de DOS: `set path=%path%;C:\python27`
y luego ya se puede usar

`python`

Usando el intérprete de Python

Invocando al Interpreter

```
C:\Users\Javier Roa>python
Python 2.7 (r27:82525, Jul  4 2010, 07:43:08) [MSC v.1500 64 bit (AMD64)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Ejecutando

```
>>> 2+2
4
>>> a=5
>>> b=7
>>> a+b
12
>>>
```

Saliendo

```
>>> exit()_
```


Usando el intérprete de Python

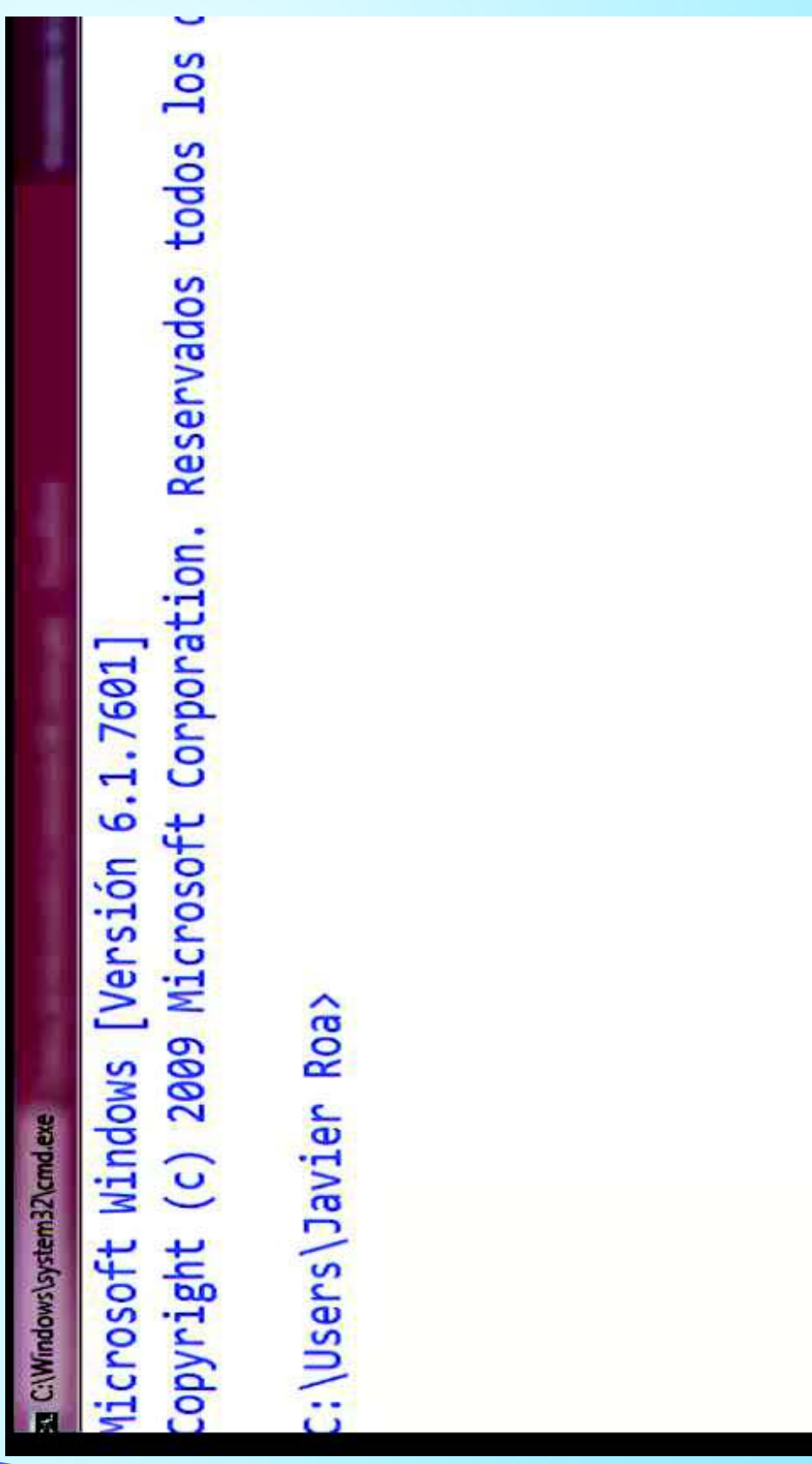
Pasando argumentos

- Cuando son conocidos por el intérprete, el nombre del script y los argumentos adicionales son entonces pasados al script en la variable `sys.argv`, una lista de cadenas de texto.
- Su longitud es al menos uno; cuando ningún script o argumentos son pasados, `sys.argv[0]` es una cadena vacía.

```
C:\Users\Javier Roa>python consulta.py 20170909
```

- En este caso los valores de `sys.argv`, serán:
 - `sys.argv[0]` → `consulta.py`
 - `sys.argv[1]` → `20170909`

HOLA MUNDO

A screenshot of a Windows command prompt window. The title bar at the top reads "C:\Windows\system32\cmd.exe". The command prompt shows the output of the 'echo' command: "Microsoft Windows [Versión 6.1.7601]", "Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.", and "C:\Users\Javier Roa>".

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.
C:\Users\Javier Roa>
```

Introducción informal a Python

Python como Calculadora

```
>>> 2+2
4
>>> # Este es un comentario... 2+2
4
>>> 2+2 # y un comentario en la misma línea que el código
4
>>> (50-5*6)/4
5
>>> # La división entera retorna redondeado al piso:
... 7/3
2
>>> 7/-3
-3
```

Introducción informal a Python

El signo igual (=) es usado para asignar un valor a una variable

```
>>> ancho = 20
>>> largo = 5*9
>>> ancho * largo
900
```

Un valor puede ser asignado a varias variables simultáneamente

```
>>> x = y = z = 0 # Cero a x, y, z
>>> x
0
>>> y
0
>>> z
0
```

Introducción informal a Python

Las variables deben estar “definidas” (con un valor asignado) antes de que puedan usarse

```
>>> # tratamos de acceder a una variable no definida
... n
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'n' is not defined
```

Se soporta completamente los números de punto flotante; las operaciones con mezclas en los tipos de los operandos convierten los enteros a punto flotante

```
>>> 3 * 3.75 / 1.5
7.5
>>> 7.0 / 2|
3.5
```

Introducción informal a Python

En el modo interactivo, la última expresión impresa es asignada a la variable "_". Esto significa que cuando estés usando Python como una calculadora de escritorio, es más fácil seguir calculando, por ejemplo:

```
>>> impuesto = 12.5 / 100
>>> precio = 100.50
>>> precio * impuesto
12.5625
>>> precio + _
113.0625
>>> round(_, 2)
113.06
```

Esta variable debería ser tratada como de sólo lectura por el usuario. No debe asignarse explícitamente un valor ; o se creara una variable local independiente con el mismo nombre enmascarando la variable con el comportamiento mágico.

Introducción informal a Python

Cadena de Caracteres:

Python puede manipular cadenas de texto, las cuales pueden ser expresadas de distintas formas. Pueden estar encerradas en comillas simples o dobles:

```
>>> 'huevos y pan'
'huevos y pan'
>>> 'doesn\'t'
"doesn't"
>>> "doesn't"
"doesn't"
>>> "Si," le dijo.
'Si,' le dijo.
>>> "\Si,\" le dijo."
'Si,' le dijo.
>>> "Isn\'t," she said.
'Isn\'t,' she said.
```


Introducción informal a Python

Cadena de Caracteres:

Las cadenas de texto literales pueden contener múltiples líneas de distintas formas. Las líneas continuas se pueden usar, con una barra invertida como el último carácter de la línea para indicar que la siguiente línea es la continuación lógica de la línea:

```
hola = "Esta es una larga cadena que contiene\n\
varias líneas de texto, tal y como se hace en C.\n\
    Notar que los espacios en blanco al principio de la línea\
son significantes."

print hola
```

```
Esta es una larga cadena que contiene
varias líneas de texto, tal y como se hace en C.
    Notar que los espacios en blanco al principio de la línea son
    significantes.
```

Introducción informal a Python

Cadena de Caracteres:

Las cadenas de texto pueden ser rodeadas en un par de comillas triples: `"""` o `'''`. No se necesita escapar los finales de línea cuando se utilizan comillas triples, pero serán incluidos en la cadena

```
print """
Uso: algo [OPTIONS]
    -h
    -H nombrehost|

Muestra el mensaje de uso
Nombre del host al cual conectarse
"""
```

```
Uso: algo [OPTIONS]
    -h
    -H nombrehost

Muestra el mensaje de uso
Nombre del host al cual conectarse
```

Introducción informal a Python

Cadena de Caracteres:

Si se hace de la cadena de texto una cadena "cruda", la secuencia \n no es convertida a salto de línea, pero la barra invertida al final de la línea y el carácter de nueva línea en la fuente, ambos son incluidos en la cadena como datos. Así, el ejemplo

```
hola = r"Esta es una larga cadena que contiene\n/  
varias líneas de texto, tal y como se hace en C."  
print hola
```

Esta es una larga cadena que contiene\n
varias líneas de texto, tal y como se hace en C.

Introducción informal a Python

Cadena de Caracteres:

Las cadenas de texto pueden ser concatenadas (pegadas juntas) con el operador + y repetidas con *:

```
>>> palabra = 'Ayuda' + 'A'
>>> palabra
'AyudaA'
>>> '<' + palabra*5 + '>'
'<AyudaAAyudaAAyudaAAyudaAAyudaAAyudaA>'
```

Dos cadenas de texto juntas son automáticamente concatenadas; esto solo funciona con dos literales, no con expresiones arbitrarias

```
>>> 'cad' 'ena'          # <- Esto es correcto
'cadena'
>>> 'cad'.strip() + 'ena' # <- Esto es correcto
'cadena'
>>> 'cad'.strip() 'ena'   # <- Esto no es correcto
Traceback (most recent call last):
...
SyntaxError: invalid syntax
```

Introducción informal a Python

Cadena de Caracteres:

Las cadenas de texto se pueden indexar; el primer carácter de la cadena tiene el índice 0(cero). Se pueden especificar subcadenas con la *notación de rebanadas*: dos índices separados por dos puntos

```
>>> palabra = 'Ayuda' + 'A'
>>> palabra
>>> palabra[4]
'a'
>>> palabra[0:2]
'Ay'
>>> palabra[2:4]
'ud'
```

Los índices de las rebanadas tienen valores por defecto útiles; el valor por defecto para el primer índice es cero, el valor por defecto para el segundo índice es la longitud de la cadena a rebanar

```
>>> palabra[:2]      # Los primeros dos caracteres
'Ay'
>>> palabra[2:]      # Todo menos los primeros dos caracteres
'udaA'
```

Introducción informal a Python

Cadena de Caracteres:

Los índices degenerados en las rebanadas son manejados bien: un índice muy largo es reemplazado por la longitud de la cadena, un límite superior más chico que el límite menor retorna una cadena vacía

```
>>> palabra = 'Ayuda' + 'A'
>>> palabra
>>> palabra[1:100]
'yudaA'
>>> palabra[10:]
''
>>> palabra[2:1]
''
```

Los índices pueden ser números negativos, para empezar a contar desde la derecha.

```
>>> palabra[-1] # El último carácter
'A'
>>> palabra[-2] # El penúltimo carácter
'a'
>>> palabra[-2:] # Los últimos dos caracteres
'aA'
>>> palabra[:-2] # Todo menos los últimos dos caracteres
'Ayud'
```

Introducción informal a Python

Cadena de Caracteres:

Una forma de recordar cómo funcionan las rebanadas es pensar en los índices como puntos *entre* caracteres, con el punto a la izquierda del primer carácter numerado en 0. Luego, el punto a la derecha del último carácter de una cadena de n caracteres tienen índice n , por ejemplo

+	-	+	-	+	-	+	-	+	-	+	-	+	-	+
	A		y		u		d		a		A			
+	-	+	-	+	-	+	-	+	-	+	-	+	-	+
0	1	2	3	4	5	6								
-6	-5	-4	-3	-2	-1									

La primer fila de números da la posición de los índices 0...6 en la cadena; la segunda fila da los correspondientes índices negativos. La rebanada de i a j consiste en todos los caracteres entre los puntos etiquetados i y j , respectivamente

Introducción informal a Python

Cadena de Caracteres:

La función incorporada `len()` devuelve la longitud de una cadena de texto:

```
>>> s = 'supercalifrastrilisticoespialidoso'
>>> len(s)
33
```

Introducción informal a Python

Variables

- No necesitan ser inicializadas
- La asignación de un valor a una variable define su tipo
- Al asignarle un valor automáticamente se convierte en un objeto (con atributos y métodos)

```
>> a='hola mundo'
>> type(a)
type 'str'>
>> b=12
>> type(b)
type 'int'>
>> c=1.3
>> type(c)
type 'float'>
>> d=[]
>> type(d)
type 'list'
```

Introducción informal a Python

Variables

Las variables cuando reciben una asignación se convierten en un objeto del tipo de valor asignado y adquieren métodos de este tipo, los métodos pueden visualizarse con la función `dir(variable)`

Ejemplo de una variable de tipo string

```
>>> dir(a)
['_add_', '__class__', '__contains__', '__delattr__', '__doc__',
'_eq_', '_format_', '_ge_', '_getattribute_', '_getitem_', '_getnewargs_',
'_get',
'_slice_', '_gt_', '_hash_', '_init_', '_le_', '_len_',
'_lt_', '_mo',
'd_', '_mul_', '_ne_', '_new_', '_reduce_', '_reduce_ex_',
'_repr_',
'_rmod_', '_rmul_', '_setattr_', '_sizeof_', '_str_',
'_subclasshook',
'_formatter_field_name_split', '_formatter_parser', 'capitalize',
'center',
'count', 'decode', 'encode', 'endswith', 'expandtabs', 'find', 'format',
'index',
'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace', 'istitle',
'isupper',
'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace', 'rfind',
'rindex',
'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines',
'startswith',
'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
>>>
```

MODULOS (clase1)

Para escribir un programa se debe preparar la entrada para el interprete y ejecutarlo con ese archivo como entrada. Esto es conocido como crear un *guión*, o *script* o *fuentes*.

Para soportar esto, Python tiene una manera de poner definiciones en un archivo y usarlos en un script o en una instancia interactiva del intérprete.

Tal archivo es llamado **módulo**; las definiciones de un módulo pueden ser *importadas* a otros módulos o al módulo *principal*. Un módulo es un archivo conteniendo definiciones y declaraciones de Python. El nombre del archivo es el nombre del módulo con el sufijo **.py** agregado. Dentro de un módulo, el nombre del mismo (como una cadena) está disponible en el valor de la variable global `__name__`.

MODULOS

```
pythonModulo.py x
1 #Modulo ejemplo de Python
2 """
3 Modulo que solicita el nombre de una persona y lo saluda
4 """
5 nombre=raw_input('Ingrese su nombre: ')
6 print "Hola " , nombre.upper(), ' como estas?'
7 print '\n'
8 raw_input('<enter> Para continuar')
9
```

```
pythonModulo pythonModulo
C:\Python27\python.exe "C:/ARTEMAR/UAJ/JIT-CITA 2018/pythonModulo.py"
Ingrese su nombre: javier roa benitez
Hola JAVIER ROA BENITEZ como estas?

<enter> Para continuar
```

Herramientas de control de Flujos y mas...

PALABRA	UTILIZACIÓN	Python
#	Comentario una línea	#
"""	Inicio/Fin de comentario Largo	"""
=	Asignación	=
LEER	Leer un dato del teclado	raw_input('mensaje')
IMPRIME	Visualiza datos en una pantalla	print 'hola mundo'
Y	Conjunción Lógica	and
O	Disyunción lógica	or
NO	Niega la condición que le sigue	not
SI	Inicia la selección SI	If (condición):
SI_NO	Complemento opcional de la selección SI	Else:
CASO	Selección entre múltiples alternativas	elif (condición):
PARA	Inicia un número fijo de iteraciones	for x in range(1,100):
MIENTRAS	Inicia la iteración MIENTRAS(condicional)	while (condición):

Herramientas de control de Flujos y mas...

De uso General:

Comentario
una linea

Comentario líneas
multiples

Solicitud de datos por
teclado

```
#Modulo ejemplo de Python
```

```
"""
```

```
Modulo que solicita el nombre de una persona y lo saluda
```

```
"""
```

```
nombre=raw_input('Ingrese su nombre: ')
```

```
x=20
```

```
print '-'*x
```

```
print "Hola " , nombre.upper(), ' como estas?'
```

```
print '-'*x
```

```
print '\n'
```

```
raw_input('<enter> Para continuar')
```

Asignacion

Impresion

Herramientas de control de Flujos y mas...

Condicionantes : (if..else..elif)

Para la toma de decisión por una condición se utiliza la palabra clave **if** , en la forma :

```
if (condición):  
    print "se cumple la condición"
```

Si es necesario ejecuta otra acción por el incumplimiento de la condición se combina con **else** , en la forma :

```
if (condición):  
    print "se cumple la condición"  
else:  
    print "no se cumple la condición"
```

***** En Python no se cierra el if , NO existe un endif *****

Herramientas de control de Flujos y mas...

Condicionantes : (if..else..elif)

Los operadores de condición son:

== (igual), **>** (mayor que), **<** (menor que) y sus combinaciones

Los operadores lógicos:

and (y), **or** (o), **not** (negación)

```
>>>a=5
>>> if a==5:
...     print 'se cumple la condicion'
... else:
...     print 'no se cumple la condicion'
...
se cumple la condicion
>>>a=2
>>>if (condición):
...     print "se cumple la condición"
>>> if a==5:
...     print 'se cumple la condicion'
... else:
...     print 'no se cumple la condicion'
...
no se cumple la condicion
>>>
```

Herramientas de control de Flujos y mas....

Condicionantes : (if..else..elif)

Puede haber cero o más bloques **elif**, y el bloque **else** es opcional. La palabra reservada **'elif'** es una abreviación de **'else if'**, y es útil para evitar un sangrado excesivo. Una secuencia **if ... elif ... elif ...** sustituye las sentencias **switch** o **case** encontradas en otros lenguajes.

```
>>> x = int(raw_input("Ingresa un entero, por favor: "))
Ingresa un entero, por favor: 42
>>> if x < 0:
...     x = 0
...     print 'Negativo cambiado a cero'
...     elif x == 0:
...         print 'Cero'
...     elif x == 1:
...         print 'Simple'
...     else:
...         print 'Mas'
...
'Mas'
```

Herramientas de control de Flujos y mas...

Repetitivas: for....

La sentencia **for** de Python itera sobre los ítems de cualquier secuencia (una lista o una cadena de texto), en el orden que aparecen en la secuencia. Por ejemplo:

```
>>> a='hola mundo'
>>> for i in a:
...     print i|
...
h
o
l
a

m
u
n
d
o
>>>
```

Herramientas de control de Flujos y mas....

Repetitivas: for....

La función `range()`

Si se necesita iterar sobre una secuencia de números, es apropiado utilizar la función integrada `range()`. Esta genera una lista conteniendo progresiones aritméticas.

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

El valor final dado nunca es parte de la lista; `range(10)` genera una lista de 10 valores, los índices correspondientes para los ítems de una secuencia de longitud 10. Es posible hacer que el rango empiece con otro número, o especificar un incremento diferente (incluso negativo; algunas veces se lo llama 'paso'):

```
>>> range(5, 10)
[5, 6, 7, 8, 9]
>>> range(0, 10, 3)
[0, 3, 6, 9]
>>> range(-10, -100, -30)
[-10, -40, -70]
```

Herramientas de control de Flujos y mas....

Repetitivas: for....

Un ejemplo de la combinación **for** y **range()** seria:

```
>>> a = ['Mary', 'tenia', 'un', 'corderito']
>>> for i in range(len(a)):
...     print i, a[i]
...
0 Mary
1 tenia
2 un
3 corderito
```

Herramientas de control de Flujos y mas...

Repetitivas: while

La sentencia **while**, es una sentencia de repetición, es lo que en pseudocódigo esta representado por (mientras p para), su forma es:

```
a=0
while a<>99:
    print a, ' No es 99'
    a=raw_input('Valor de a: '), a
```

También puede usarse con el valor lógico True, así:

```
while True:
    a=raw_input('Valor de a: ')
    if not a:
        break
    else:
        print 'El valor de a es: ', a
```

Ejercicios



Ejercicios

1. Hacer un programa que permita ingresar dos números, y muestre la multiplicación de ambos
2. Hacer un programa que permita ingresar tres números, y muestre en pantalla tanto la suma como la multiplicación de ellos
3. Confeccione un programa que lea un número e indique si este es positivo o negativo
4. Confeccione un programa que lea un número e indique si este es par o impar
5. Confeccione un programa que lea un número e indique si este es par-positivo, par-negativo, impar-positivo o impar-negativo.
6. Confeccione un programa que lea un número y si este es mayor o igual a 10 devuelva el triple de este, de lo contrario la cuarta parte de este.
7. Obtener el IVA de una venta y si esta es superior a G 150.000 aplicar un descuento del 25 %.