Tipos de Datos

Tipos de Datos - Temario

Repaso Clase Anterior

Variables y Tipos de Datos

Tipos de Datos - Bibliografía

Aprender a Pensar como un Programador en Python

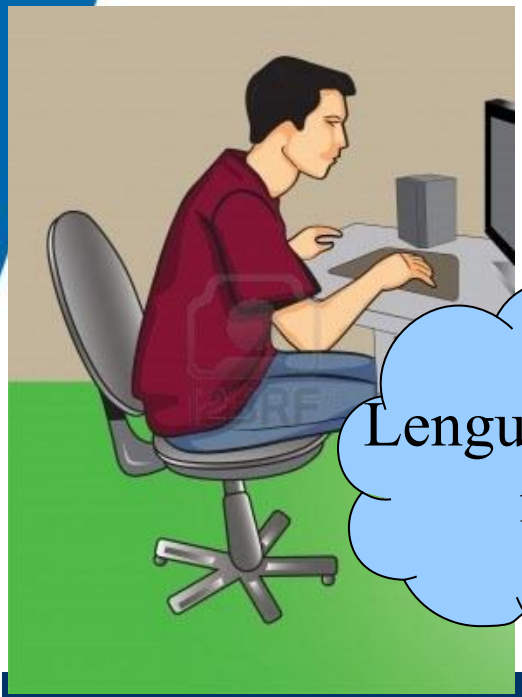
Capítulo 2

Repaso Clase Anterior

Repaso Clase Anterior

Programa

Conjunto de órdenes que explica a la computadora cómo realizar una tarea.



```
numero=input('Ingresa un numero')
numero= int(numero)+3
```

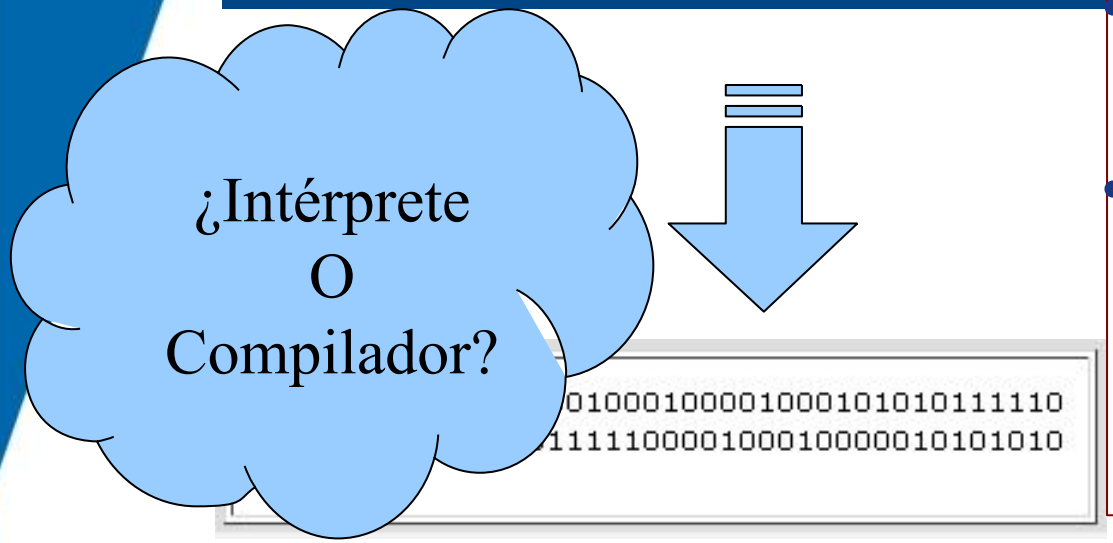
Lenguaje de Alto nivel

```
0001010001000100010001000010
0000011101010001111100001000
```

Lenguaje de máquina

Repaso Clase Anterior

```
numero=input('Ingresa un numero')
numero= int(numero)+3
```



- Lenguajes fáciles de escribir
- Permiten programas portables
- Necesitan ser traducidos a lenguaje de máquina

Concepto de variable

Antes de empezar...

```
print("Hola")
print (2 +1)
print (3*3)
```

…..

"Hola", 2, 1, 3,-- son **valores.**

El **valor** es uno de los elementos fundamentales que manipula un programa.

Variable

Una **variable** es un nombre que representa o refiere a un **valor**. Ejemplo:

```
>>> x = 3
```

El nombre "**x**", representa el valor 3

Una variable sirve para generalizar los valores.
Su uso es similar al que le damos en matemáticas
Pueden sustituirse por cualquier valor.

En Python las variables no se declaran.
Simplemente, se usan.

Variable

El nombre de las variables pueden contener letras, dígitos y "_" . Deben comenzar con letra.

MiVariable
MiVar1
mi_var1

¡¡Correctos!!

1MiVariable
"miVar"
mi_var**
Mi var

¡¡Incorrectos!!

Importante:

Hay que asignarle un valor a una variable **antes** de poder utilizarla.

En Python **HAY** diferencia entre mayúsculas y minúsculas: variable **miVar** es distinto de variable **MiVar.**

No pueden usarse palabras reservadas como nombre

Variable

Algunos nombres tienen un significado especial

- Ejemplo: aquellos que empiezan con “__”

La PEP8 (Python Enhancement Proposals), define algunas sugerencias en la codificación.

- No usar “l”, “i”, “o” como nombres de variables
- Nombres de variables siempre en minúsculas
- Usar “_” si los nombres son largos.
- Ejemplo: mi_nueva_var

http://www.python.org/dev/peps/pep-0008/

Variable

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

Las variables DEBEN tener un valor ANTES de usarse

Variable

```
x = 10
print(x)
```

Sentencia de Asignación.

La variable x se refiere al valor 10.

Comentarios

Cuando escribimos programas es importante **documentar** lo que hacemos.

Los comentarios NO forman parte de las instrucciones que ejecuta la máquina.

Son importantes para las personas que leen y mantienen los programas.

Hasta ahora...

Para insertar un comentario se utiliza el símbolo '**#**'.

Si el comentario ocupa varias líneas, debe repetirse '**#**' en cada línea.

Ejemplo

```
#Inicializamos las variables
x=10
y=20
#Calculamos la suma
z=x+y
#Imprimimos el resultado
print(z)
```

Tipos de Datos

Tipo de datos

Definición:

Un Tipo de datos define un conjunto de valores y las operaciones válidas que pueden realizarse sobre esos valores

- Conjunto de valores:
  - Representa **todos los valores posibles** que puede llegar a tomar una variable de ese tipo
- Operaciones permitidas:
  - Establece qué operaciones son válidas para los datos pertenecientes a dicho tipo

Tipo de datos

Básicos:

- Números
  - Enteros
  - Flotantes
- Booleanos
- Cadenas de texto

Colecciones (Estructuras de datos)

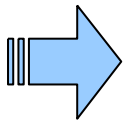
- Listas
- Tuplas
- Conjuntos
- Diccionarios

Tipos Básicos - Enteros

Permite trabajar con valores enteros negativos y positivos.

Se representan mediante los tipos **int**

```
x=10
type(x)
```



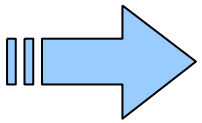
```
>>> x=10
>>> type(x)
<type 'int'>
>>>
>>>
```

Se almacenan en 4 u **8 bytes**

Tipos Básicos - Reales

Permite trabajar con valores con coma decimal.
Se representan mediante el tipo **float.**

```
x=10.5
type(x)
```



```
>>>
>>> x=10.5
>>> type(x)
<type 'float'>
>>>
>>>
```

Se almacenan en **64 bits = 8 bytes**

Operadores aritmético

Operaciones que pueden hacerse sobre variables numéricas y números.

	Operador	Descripción
Operadores aritméticos	+	Suma
	-	Resta
	*	Multiplicación
	/	División
	-	Negación
	**	Exponente
	//	División entera
	%	Resto de la división

¿Qué sucede si...?

En una expresión se combinan distintos tipos de datos:

Ejemplo: print(10.5/2)

La división entre enteros nos entrega un resultado del tipo Flotante.

i= 7/2, da como resultado 3.5

Una expresión con números mixtos **se convierte** a flotante.

i= 7.5/2, da como resultado 3.75

Existen funciones que realizan **conversiones explícitas**, como por ejemplo: **float() e int()**.

i= int(7.5/2), da como resultado 3

En python2 al dividir dos números enteros el resultado se "redondea" para abajo.

Tipos Básicos - String

No todos son números....
Usamos cadenas de caracteres para valores que representan:

- Nombres de personas, países, ciudades
- Direcciones postales, de mail,
- Mensajes,
- Etc.

Ejemplos:

"Juan Pérez"; "Argentina"; "Florencio Varela";
"juan.perez@gmail.com", "Hola que tal"

Tipos Básicos - String

Secuencia de caracteres (letras, números, marcas de puntuación, etc.)

Se encierran entre comillas simples '' o comillas dobles “ ”

Algunos operadores:

+ **Concatenación**

* **Repetición**

```
>>> saludo='Hola'
>>> print(saludo + "Python")
HolaPython
```

```
>>> saludo='Hola'
>>> print(saludo*3)
HolaHolaHola
```

Tipos Básicos - String

Operadores de comparación: **==**, **!=**, **>**, **<**, **>=**,**<=**

Ejemplos:

```
>>>  'pepe ' == 'pepe'
true
>>> “juan”<”ana”
false
```

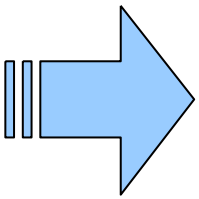
Python utiliza un criterio de comparación de cadenas muy natural: **el orden alfabético.**

Python utiliza los códigos ASCII de los caracteres para decidir su orden

Tipos Básicos - String

Para saber el orden que ocupa un carácter se cuenta con las funciones predefinidas **“ord()”** y **“chr()”**, su función inversa.

```
>>> "N"<"a"
True
>>> ord("N")
78
>>> ord("a")
97
```



Notar que:
'N'<'a'!!!

Tipos Básicos - String

Funciones predefinidas que manipulan cadenas:

Funciones	Descripción	Ejemplo	Resultado
int	Convierte una cadena numérica a un valor entero	int("123")	123
float	Convierte una cadena numérica a un valor real	float("123")	123.0
ord	Devuelve el código ASCII de la cadena	ord("a")	97
chr	Devuelve el carácter correspondiente al valor ASCII	chr("89")	"T"
str	Convierte un valor entero a una cadena de caracteres	str(123)	"123"

Tipos Básicos - String

Otras cosas útiles....

Funciones	Descripción	Ejemplo	Resultado
a.lower()	Convierte los caracteres de la cadena a a minúsculas	pal="HOLA" print pal.lower()	"hola"
a.upper()	Convierte los caracteres de la cadena a a mayúsculas	pal="hola" print pal.upper()	"HOLA"

Tipos Básicos - String

Longitud de las cadenas

Uso de función predefinida **len()**

```
mensaje="Hola a todos"
print("el mesnaje tiene", len(mensaje), " caracteres")
```

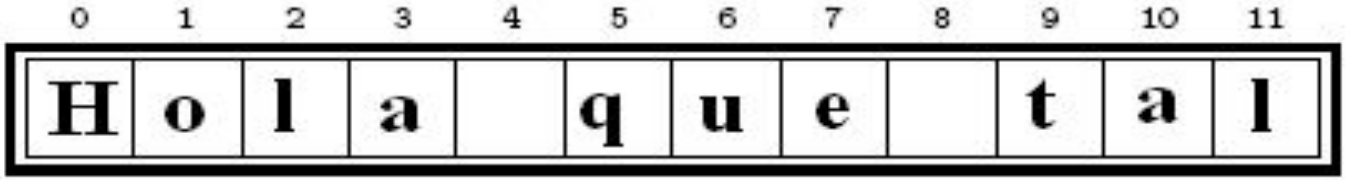
len('') devuelve longitud **0**

len(' ') devuelve longitud **1**

Tipos Básicos - String

Accediendo a los caracteres de las cadenas

cadena = 'Hola que tal'




```
>>> mensaje="hola que tal"
>>> print(mensaje[0])
h
>>> print(mensaje[len(mensaje)-1])
l
```

len(mensaje): cantidad de caracteres

Pero.... inicia en 0

Tipos Básicos -Boolean

Tipo de dato con solo dos valores posibles:
"Verdadero" o "Falso"

Al tipo de dato **Boolean**

Valores booleanos: True y False
Operadores lógicos: and, or, not....

Condiciones y resultados….

Ejemplos:

a=(3>1) → print(a) → True

b=(3>3) → print(b) → False

Operadores lógicos:

and	True	False
True	**True**	False
False	False	False

or	True	False
True	True	True
False	True	**False**

Estructura de un programa

Sentencia: orden ejecutable que compone un programa.

Flujo de ejecución: forma en que se ejecutan las sentencias.

Estructuras de control:

Son el medio por el cual los programadores pueden determinar el flujo de ejecución en un programa

Secuencias de instrucciones

Si se debe calcular el sueldo con el 20% de aumento de un empleado...

Con el intérprete

```
>>> antiguedad = input('Ingrese su antiguedad en el trabajo: ')
Ingrese su antiguedad en el trabajo: 10
>>> sueldo = input('Ingrese su sueldo: ')
Ingrese su sueldo: 10000
>>> sueldo = int(sueldo) + int(antiguedad) * 0.20
>>> print(sueldo)
10002.0
```

El flujo de ejecución es **"secuencial".** Se ejecuta una sentencia y luego la siguiente

....Pero .. ¿Si el aumento hubiese sido SOLO para los que tenían 10 o más años de antigüedad?

Se debería haber usado una estructura de control que permitiese testear si la antigüedad era mayor o igual que 10