ESCUELA POLITÉCNICA NACIONAL

cec·epn CENTRO DE EDUCACIÓN CONTINUA

UNIDAD 2

Tipos de Datos, variables, operaciones básicas de entrada y salida, operadores básicos

Valores y Tipos de Datos.

Un “valor” es un dato, como parte fundamental en la operación de un programa, como letras o números.

Los valores pertenecen a diferentes tipos o *class*: 8 es un entero o *int*, y "¡Hola mundo!" es una cadena o *str*. A través del intérprete se puede identificar los tipos de datos mediante una función llamada *type()*, y realizar comparaciones de valor mediante la función *isinstance()*.

```
>>> type(3)
<class 'int'>
>>> type(2.78)
<class 'float'>
>>> type('Hola')
<class 'str'>
>>> isinstance(3, float)
False
>>> isinstance(3, int)
True
>>> isinstance(3, bool)
False
>>> isinstance(False, bool)
True
```

Tipos de Datos Simples.

Datos Numéricos: Permiten representar valores de forma numérica, esto incluye a los números enteros y los reales. Permiten realizar operaciones aritméticas comunes.

45, 3.5692, 5+7i

Datos Lógicos: Aquellos que solo pueden tener dos valores ya que representan el resultado de una comparación entre otros datos.

True, False

Datos Alfanuméricos (String): Es una secuencia de caracteres alfanuméricos que permiten representar valores identificables de forma descriptiva. Es posible representar números como alfanuméricos, pero estos pierden su propiedad matemática. Este tipo de datos se representan encerrados entre comillas (python : comillas simples o dobles)

“Centro de Educacion Continua” , “A”, “2021”

Variables.

Son objetos que reside en la memoria donde se guarda y se recuperan datos que se utilizan en un programa. Se utilizan para:

- Guardar datos y estados.
- Asignar valores de una variable a otra.
- Representar valores dentro de una expresión matemática.
- Mostrar valores por pantalla.

Todas las variables deben ser de un tipo de dato específico. Se debe usar el símbolo de asignación de valor =.

```
>>> a=6
>>> texto="Hola mundo"
>>> a1=98
>>> numero1=4.5
>>> numero2=4,5
>>> numero2=4.6
```

Entrada por teclado.

La función input() permite obtener texto escrito por teclado. Al llegar a la función, el programa se detiene esperando que se escriba algo y se pulse la tecla Intro.

```
input()
```

Variables.

Convenciones de nombres para las variables:

- Nunca use símbolos especiales como !, @, #, $, %, etc.
- El primer carácter no puede ser un número.
- Los nombres pueden tener la combinación de letras en minúsculas o MAYÚSCULAS (de la A a la Z) o dígitos (del 0 al 9) o un underscore (_). Por ejemplo:
- Los nombres que comienzan con guión bajo (simple _ o doble __) se reservan para variables con significado especial
- No pueden usarse como identificadores, las palabras reservadas .

Python Keywords

False	def	if	raise
None	del	import	return
True	elif	in	try
and	else	is	while
as	except	lambda	with
assert	finally	nonlocal	yield
break	for	not	continue
class	from	or	
	global	pass	

Expresiones.

Una expresión es una combinación de valores, variables y operadores. Al digitar una expresión en la línea de comandos, el intérprete la evalúa y despliega el resultado

Una expresión consta de “operadores” y “operandos”. Según sea el tipo de datos que manipulan, se clasifican en:

- **Aritméticas**. Ejemplo, 2 + 3 es una expresión aritmética que, al ser evaluada, siempre entrega el valor 5 como resultado. En esta expresión, 2 y 3 son valores literales y + es el operador de adición.

- **Relacionales**, Ejemplo: a<b

- **Lógicas**, Ejemplo: not a

Una expresión puede ser usada como una sentencia de un programa por sí sola, pero la mayoría de las veces esto no tiene ningún efecto. El programa evaluará la expresión, pero no hará nada con el resultado obtenido.

Operadores.

Son elementos que relacionan de forma diferente, los valores de una o más variables y/o constantes. Es decir, los operadores permiten manipular valores.

Operadores Aritméticos*:* Permiten la realización de operaciones matemáticas con los valores. Los operadores aritméticos pueden ser utilizados con tipos varios de datos.

+	Suma	
-	Resta	
*	Multiplicación	
/	División	
%	Módulo	(resto de la división entera)
//	Cociente	

Prioridad

- Todas las expresiones entre paréntesis se evalúan primero. Las expresiones con paréntesis anidados se evalúan de dentro a fuera.
- Dentro de una misma expresión los operadores se evalúan en el siguiente orden.

1. ** Exponenciación
2. *, /, % Multiplicación, división, módulo.
3. +, - Suma y resta.

- Los operadores en una misma expresión con igual nivel de prioridad se evalúan de izquierda a derecha.

Operadores de texto*:* Los operadores + y * tienen otras interpretaciones cuando sus operandos son strings.

+ es el operador de concatenación de strings: pega dos strings uno después del otro, no es una suma, Ni una operación conmutativa.

```
>>> 'perro' + 'gato'
'perrogato'
```

* es el operador de repetición de strings. Recibe un operando string y otro entero, y entrega como resultado el string repetido tantas veces como indica el entero.

```
>>> 'waka' * 2
'wakawaka'
```

Declaraciones.

Son instrucciones lógicas que el intérprete de Python puede leer y ejecutar. Puede ser una expresión o una instrucción de asignación.

```
1 >>> 2 + 3
2 5
3 >>> x = 2 + 3
```

Una expresión es una declaración de Python, porque el intérprete de Python puede leerla y ejecutarla

Declaración de asignación: Son la base de Python. Define cómo se crean y guardan objetos.

```
1 >>> x = x + y
```

Declaración de asignación mejorada: Se puede combinar operadores aritméticos en la asignación para formar una declaración de asignación mejorada.

Asignación aumentada:	es equivalente a:
a += b	a = a + b
a -= b	a = a - b
a *= b	a = a * b
a /= b	a = a / b
a **= b	a = a ** b
a //= b	a = a // b
a %= b	a = a % b

Manejo de números.

Son tipos de datos que se crean mediante valores numéricos y se devuelven como resultados por operadores aritméticos y funciones aritméticas integradas. Los objetos numéricos son inmutables.

Los números de Python están fuertemente relacionados con los números matemáticos, pero están sujetos a las limitaciones de la representación numérica en las computadoras.

Existen 3 tipos básicos de datos numéricos:

int – enteros
float – con decimales
complex – complejos

Enteros.

En Python se pueden representar mediante el tipo int. Según la arquitectura del procesador, Python puede almacenar números según:

En 32 bits: de -2.147.483.648 a 2.147.483.647.
En 64 bits, de -9.223.372.036.854.775.808 hasta 9.223.372.036.854.775.807.

```
1  #Variables de valor entero
2  x = 9
3  y = 64895
4  z = -325
```

Decimales.

Este tipo de dato se representa en lenguaje de programación como float (flotante). Puede, al igual que el entero, ser positivo o negativo, conteniendo uno o más decimales.

```
11  x = 1.10
12  y = 1.0
13  z = -35.59
14
15  print(x)
16  print(y - z)
```

La variable *float* también acepta números en notación científica, en los cuales se coloca una «e» para indicar el valor de la potencia base 10.

```
20  #Variables con potencias
21  x = 35e3
22  y = 12e4
23  z = -87.7e100
```

En Python los números decimales se almacenan internamente en binario con 53 bits de precisión (IEEE-754). Cuando un programa pide a Python un cálculo con números decimales, Python convierte esos números decimales a binario, realiza la operación en binario y convierte el resultado de nuevo en decimal para mostrárselo al usuario.

El problema es que muchos números decimales no se pueden representarse de forma exacta en binario, por lo que los resultados no pueden ser exactos.

```
>>> round(3.45, 1)
3.5
>>> round(3.55, 1)
3.5
```

En el primer valor, al convertir 3.45 a binario con 53 bits de precisión, el valor obtenido es realmente 3.450000000000001776356839400250464677810668945312 5, es decir, ligeramente mayor que 3.45, por lo que al redondear con décimas, Python muestra el valor 3.5.

En el segundo valor, al convertir 3.55 a binario con 53 bits de precisión, el valor obtenido es realmente 3.549999999999999822364316059974953532218933105468 75, es decir, ligeramente inferior a 3.55, por lo que al redondear con décimas, Python muestra también el valor 3.5.

```
>>> from decimal import Decimal
>>> Decimal(3.45)
Decimal('3.450000000000001776356839400250464677810668945312 5')
>>> Decimal(3.55)
Decimal('3.549999999999999822364316059974953532218933105468 75')
```

Si en una aplicación concreta se necesita total exactitud, se deben utilizar bibliotecas específicas. Python incluye la biblioteca “decimal” y existen bibliotecas como “mpmath” que admiten precisión específica.

Complejos.

Los números complejos o imaginarios se escriben con una «j» como la parte imaginaria, siempre acompañada de un número. Es posible operarlos a través de expresiones o declaraciones. Su clase se define como *complex*.

```
25  #Variables con números imaginarios
26  x = 5j
27  y = 3 + 7j
28  z = 9 - 5j
29
30  print(x)
31  print(y + z)
```

Una vez creado, es posible acceder a la parte real con *real* y a la imaginaria con *imag.*

```
c = 3 + 5j
print(c.real) #3.0
print(c.imag) #5.0
```

También se puede crear un número complejo haciendo uso de complex, pero sin usar la j.

```
c = complex(3,5)
print(c) #(3+5j)
```

Cadenas de Caracteres - Strings

Los cadenas (o strings) son un tipo de datos compuestos por secuencias de caracteres que representan texto. Estas cadenas de texto son de tipo str y se delimitan mediante el uso de comillas simples o dobles.

```
>>> cadena = "Programa en Python"
>>> type(cadena)
<class 'str'>
```

Cuando se necesita usar comillas sencillas o dobles, dentro de una cadena, se tienen distintas opciones. La más simple es encerrar la cadena mediante un tipo de comillas (simples o dobles) y usar el otro tipo dentro de la cadena.

```
>>> print("Decimos 'si' entre comillas")
Decimos 'si' entre comillas
```

Otra opción es usar en todo momento el mismo tipo de comillas, pero usando la barra invertida (\) como carácter de escape en las comillas del interior de la cadena para indicar que esos caracteres forman parte de la cadena.

```
>>> print('Decimos \'si\' entre comillas')
Decimos 'si' entre comillas
```

La función len(), indica el número de elementos de un objeto. En el caso de que el objeto sea una cadena nos indica el número de caracteres que la componen.

```
>>> cadena = "Programa en Python"
>>> len(cadena)
18
```

Indexación de Strings

Cada uno de los caracteres de una cadena incluidos los espacios tiene asignado un índice. Este índice nos permite seleccionar su carácter asociado haciendo referencia a él entre corchetes ([]) en el nombre de la variable que almacena la cadena. Si consideremos el orden de izquierda a derecha, el índice empieza en 0 para el primer carácter.

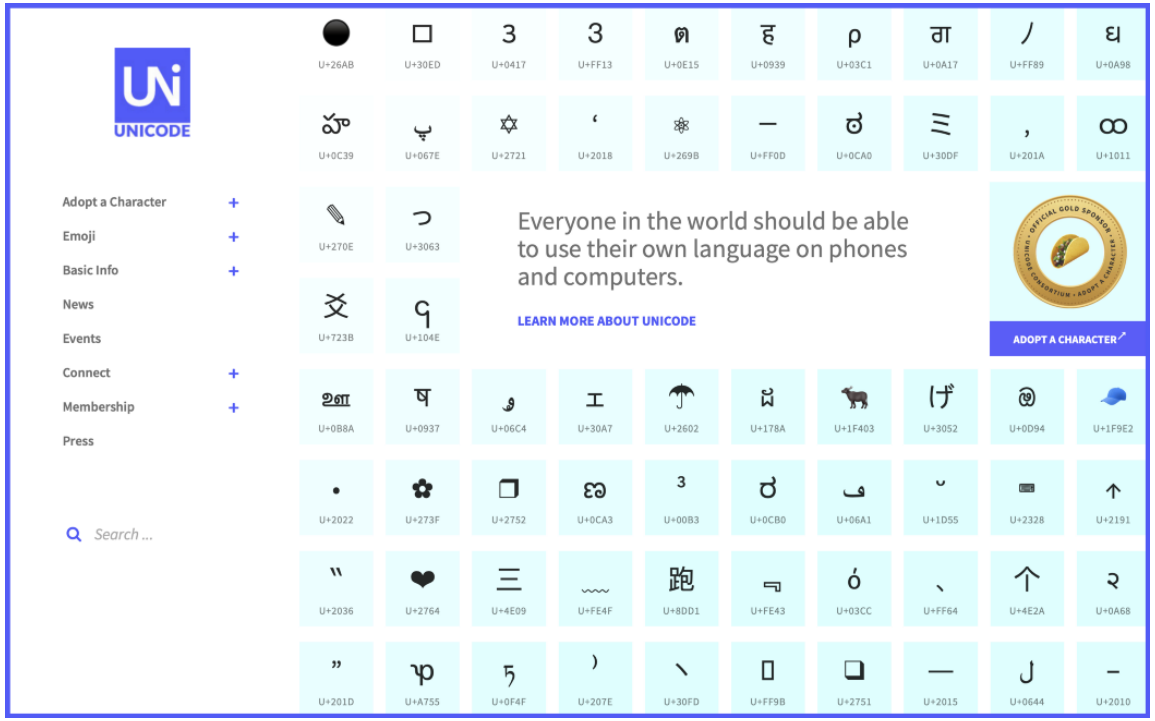
Caracteres :	P	y	t	h	o	n
Índice :	0	1	2	3	4	5
Índice inverso :	-6	-5	-4	-3	-2	-1

También se puede considerar el orden de derecha a izquierda, en cuyo caso al último carácter le corresponde el índice -1, al penúltimo -2 y así sucesivamente.

```
>>> cadena = "Programa en Python"
>>> cadena[0]
'P'
>>> cadena[-1]
'n'
```

Codificación de Strings

Las cadenas de str se manejan internamente como una secuencia de puntos de código Unicode. UTF – Unicode Transformation Format.



Las codificaciones más comunes para los idiomas occidentales son UTF-8 y UTF-16, que usan secuencias de valores de uno y dos bytes respectivamente para representar cada punto de código.

Cadenas f

En Python 3.6 una nueva notación para cadenas llamada cadenas "f", que simplifica la inserción de variables y expresiones en las cadenas. Una cadena "f" contiene variables y expresiones entre llaves ({}) que se sustituyen directamente por su valor. Las cadenas "f" se reconocen porque comienzan por una letra f antes de las comillas de apertura.

```
nombre = "Pepe"
edad = 25
print(f"Me llamo {nombre} y tengo {edad} años.")
```

Cadenas sin formato

Es una cadena con el prefijo r o R . Su uso hace posible definir cadenas literales en las que las barras invertidas se tratan como caracteres, por lo que pierden la propiedad de secuencias de escape. Esta característica es particularmente útil para escribir expresiones regulares o para usarse con analizadores SQL, por ejemplo.

```
s1 = 'Hello\nWorld'
print('s1:', s1)

s2 = r'Hello\nWorld'
print('s2:', s2)
```


ESCUELA
POLITÉCNICA
NACIONAL


cec·epn
CENTRO DE EDUCACIÓN CONTINUA


python™