CURSOS INTERSEMESTRALES

PROTECO

Tipos de objetos

*Amairany Solana
*Diego Casillas

Python Básico
10/ 06/2019

¿Qué es un tipo de dato?

Todos los valores de datos en Python **son objetos**, y cada objeto, o valor, **tiene "un tipo".**
Cada tipo de objeto determina qué operaciones se van a poder realizar con esos valores de los datos, qué atributos tiene.

PROTECO

¿Cómo comunicarnos con la computadora?

PROTECO

Datos de salida

Sintaxis…

print("textoQueQuieroImprimir"**)**

print() es una función que permite imprimir lo que se encuentra entre comillas en la pantalla, si ejecutamos la líne anterior, se mostraría en pantalla: ***textoQueQuieroImprimir*. Es decir, va del programa al usuario.**

PROTECO

Datos de salida

String = cadena = texto

Sintaxis…

print(“textoQueQuieroImprimir”**)**

print("textoQue",
"Quiero","Imprimir"**)**

quiero = “Quiero”
print("textoQue", **quiero**,"Imprimir"**)**
#quiero es una variable,¿Qué es una variable? Sigue adelante y lo descbrirás

PROTECO

Datos de entrada

Sintaxis…

variable **= input()**

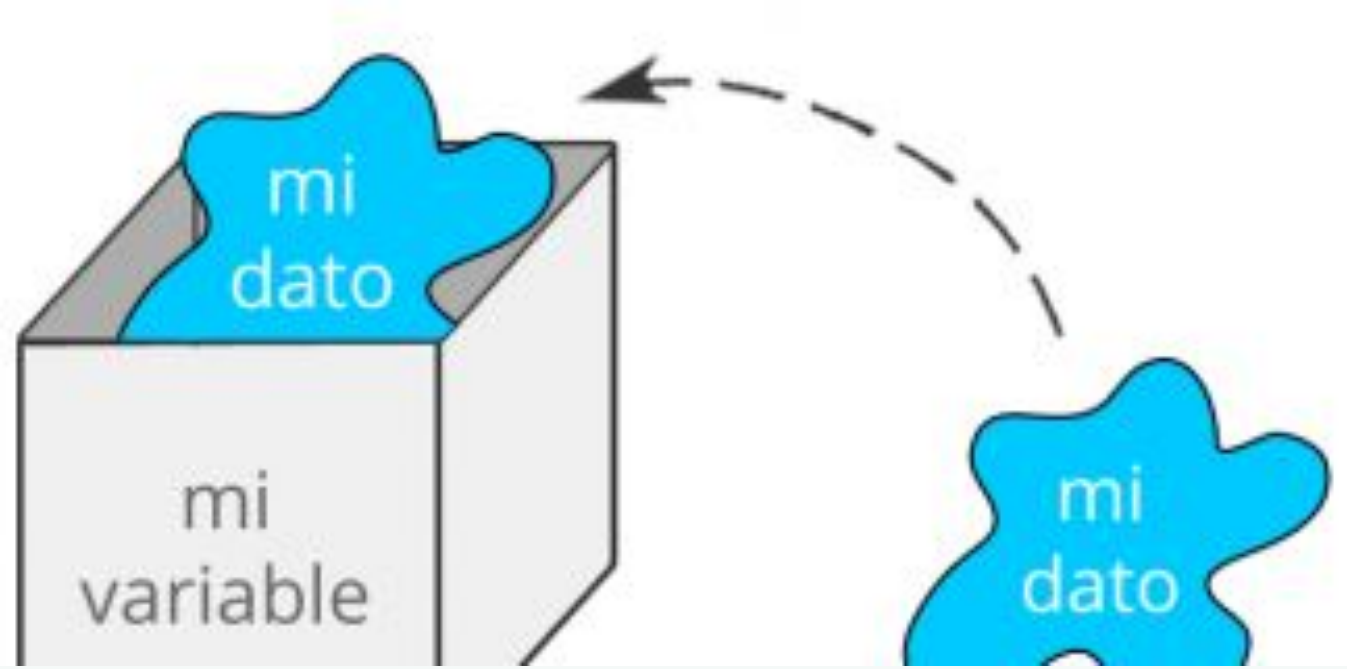
Variable **= input(**"Usuario, dame un número"**)**

os ingresados por el usuario y devolver los mismos datos al programa en ejecución.

PROTECO

Variables

Variable: Nombre o identificador asociado a un dato. Lugar donde se guardará el dato.



PROTECO

Subtemas

1. String
2. Números
3. Booleanos
4. Listas
5. Tuplas
6. Diccionarios
7. Conjuntos *

PROTECO

String

String = cadena = texto

"A string

PROTECO

Cadenas

Las cadenas son secuencias de caracteres. Un carácter representa un símbolo (letra, un número, símbolo u otra cosa). Las cadenas y los caracteres deben encontrarse entre comillas simples o dobles.

cadena = “Hola, ¿qué tal?”

cadena = ‘Hola, ¿qué tal?’

carácter = ‘H’

PROTECO

Cadenas

Las cadenas son secuencias de caracteres que mantienen un orden de izquierda a derecha y se numeran de la siguiente forma. Donde se comienza desde 0.

PROTECO

Cadenas

Acceder a un elemento de la cadena. Entre los corchetes es el elemento que se accede.

```
cadena = 'Hola, ¿qué tal?'

cadena[3]
```

El elemento seleccionado será : **a**

```
cadena[-2]
```

El elemento seleccionado será : **l**

PROTECO

String

String es un tipo de objeto (o de dato) el cual podemos utilizar.

OPERACIONES	SIGNIFICADO	EJEMPLO
string + string	Concatena dos cadenas	“Te”+ “quiero” = Te quiero
string * número número * string	Imprime el número de veces la cadena	4 * “sí” =sisisisi
string[9]	Accede al carácter 9 de la cadena llamada “string”	string = “Hola, terricolas” print(string[9]) ->r

PROTECO

String Métodos

En Python, todo es un objeto y por tanto, cualquier variable cuyo valor sea de tipo string, podrá ser tratada como un subtipo del objeto string, el cuál dispone de métodos.

Cadenas

1.- Retornar la cadena capitalizada.

```
cadena = 'hola, ¿qué tal?'
cadena.capitalize()
```

Imprime en pantalla: **'Hola, ¿qué tal?'**

2.- Retornar la cadena con un reemplazo.

```
cadena = 'hola, ¿qué tal?'
cadena.replace("tal", "hace")
```

Imprime en pantalla: **'Hola, ¿qué hace?'**

PROTECO

Cadenas

3.- Contar el número de apariciones de una subcadena en la

cadena.

```
cadena = 'hola, hola, hola'
cadena.count('o')
```

Imprime en pantalla: 3

4.- Longitud de una cadena

```
cadena = 'hola, ¿qué tal?'
len(cadena)
```

Imprime en pantalla: 15

PROTECO

Números (tipos)

CLASE	TIPO	NOTA	EJEMPLO
int	Números	Número entero.	1 3 78
long	Números	Número entero muy grande.	0261151614335
float	Números	Números fraccionales o decimales.	3.56 26.9 9.451
complex	Números	Núemros con parte real y parte imaginaria.	3 + 9j 0 + 16j 2+ 3j

PROTECO

Convertir a numéricos

FUNCIÓN	TIPO	NOTA	EJEMPLO
int()	Números	Número entero.	1 3 78
long()	Números	Número entero muy grande.	0261151614335
float()	Números	Números fraccionales o decimales.	3.56 26.9 9.451
complex()	Números	Núemros con parte real y parte imaginaria.	3 + 9j 0 + 16j 2+ 3j

PROTECO

Números (operadores)

OPERADORES EN NÚMEROS	NOTA	EJEMPLO
+	Suma	2+3 = 5 9+2 = 11
-	Resta	6-3 = 3 9-1 = 8
/	División	9/3 = 3 12/3 =4
*	Multiplicación	2*7 = 14 6*2 = 12
//	División entera (redondea)	14//4 =3
%	Módulo	14%4 =2 14//4 = 3 3*4 = 12 14-12 = 2
**	Exponencial	2**2 =4 3**3 =27

PROTECO

Números

OPERADOR LÓGICO	COMPARACIÓN	EJEMPLO
==	Es igual que	2 == 2
!=	Es distinto de	2 != 3
<	Es menor que	2 < 5
>	Es mayor que	7 > 1
<=	Es menor o igual que	2 <= 10
>=	Es mayor o igual que	9 >= 8

PROTECO

Ejercicio datos de entrada y salida

1.-Use la función **input ()** para pedir al usuario que ingrese su nombre desde la consola. Imprima el nombre en la consola.

Pista, el programa puede lucir así…

```
print("Hola, Soy Python.¿Cuál es tu nombre?")
#Completa el código
print("¡Hola, " ,nombre, " , gusto en conocerte!")
```

PROTECO

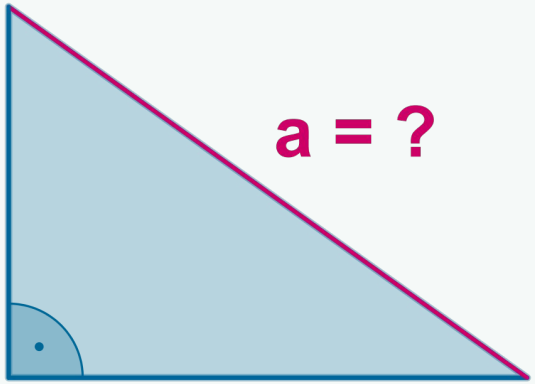
Ejercicio datos de entrada y salida

1.-Use la función **input ()** para pedir al usuario que ingrese su nombre desde la consola. Imprima el nombre en la consola.

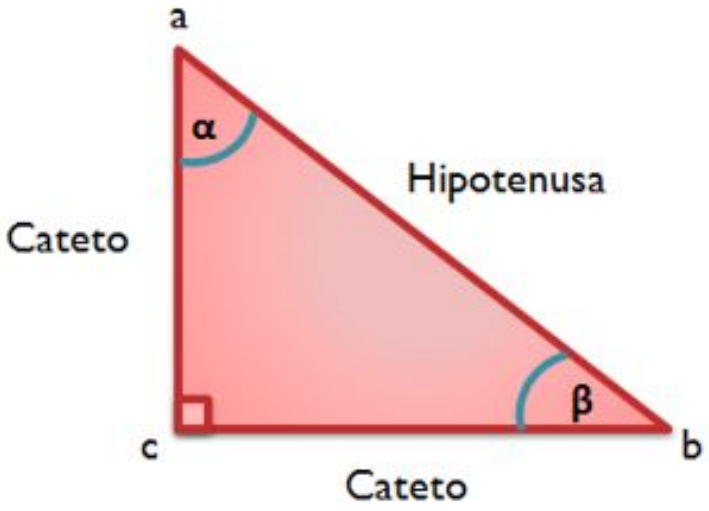
```
print("Hola, Soy Python.¿Cuál es tu nombre?")
nombre = input("Hola, Soy Python.¿Cuál es tu nombre?")
print("¡Hola, " ,nombre, " , gusto en conocerte!")

nombre = input("Hola, Soy Python.¿Cuál es tu nombre?\n")
print("¡Hola, " ,nombre, " , gusto en conocerte!")
```

PROTECO

2.-Calcula e imprime en pantalla la hipotenusa (a) del siguiente triángulo:



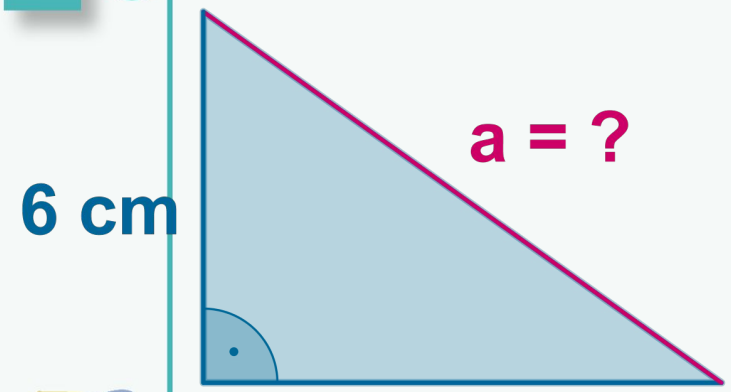
$$\text{hipotenusa} = \sqrt{\text{Cateto1}^2 + \text{Cateto2}^2}$$



PROTECO

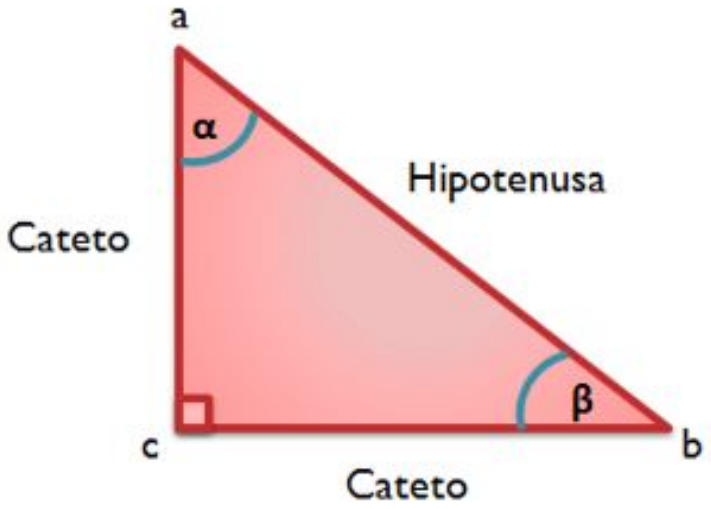
Números

```
cateto1 = 6
cateto2 = 8
hipotenusa = (cateto1**2 +
cateto2**2)**0.5
print("El valor de la hipotenusa
es:",hipotenusa)
```



PROTECO

$$\text{hipotenusa} = \sqrt{\text{Cateto1}^2 + \text{Cateto2}^2}$$



Booleanos

Una expresión booleana (o expresión lógica) es evaluada como **True** (1, verdadero) y **False** (0, falso).

En Python cualquier variable (en general, cualquier objeto) puede considerarse como una variable booleana. En general los elementos nulos o vacíos se consideran False y el resto se consideran True.

PROTECO

Booleanos

Para comprobar si un elemento se considera True o False, se puede convertir a su valor booleano
mediante la función bool().

>>> bool(0)
¿Será True o **False**?
>>> bool(4)
¿Será **True** o False?
>>> 1 == 2
False
>>> 5 == 5
True

PROTECO

Ejercicio Booleanos

Usando uno de los operadores de comparación en Python, escriba un programa simple de dos líneas que tome el parámetro **n como entrada**, que es un número entero, e imprima **False** si n es menor que 100, y **True** si n es mayor o igual que 100 .

Ejemplo de entrada: 55

Ejemplo de salida : Falso

PROTECO

Ejercicio Booleanos

Usando uno de los operadores de comparación en Python, escriba un programa simple de dos líneas que tome el parámetro **n como entrada**, que es un número entero, e imprima **False** si n es menor que 100, y **True** si n es mayor o igual que 100 .

```
n= int(input("Dame un número\n"))
print(bool(n>=100))
```

PROTECO

Listas

Las listas son estructuras de datos que nos permiten almacenar objetos de cualquier tipo: cadenas, números, listas, contenido mixto, etc.

Listas

Con la función integrada dir() podemos conocer sus métodos:

```
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__'
, '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__'
, '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__',
 '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__r
educe__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__'
, '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear',
 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort
']
>>>
```

PROTECO

Tuplas

Al igual que las listas, son estructuras de datos que almacenan objetos de cualquier tipo, incluso pueden anidar tuplas.

La diferencia: Las tuplas son inmutables.

D:D

PROTECO

Tuplas

Con la función integrada dir() podemos conocer sus métodos:

```
>>> dir(tuple)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '_
_eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs
__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__'
, '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_e
x__', '__repr__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclas
shook__', 'count', 'index']
>>>
```

PROTECO

Diccionarios

Un Diccionario es una estructura de datos y un tipo de dato en Python con características especiales que nos permite almacenar cualquier tipo de valor como enteros, cadenas, listas, etcétera .Estos diccionarios nos permiten además identificar cada elemento por una clave (Key).

PROTECO

Diccionarios

Con la función integrada dir() podemos conocer sus métodos:

```
>>> dir(dict)
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__'
, '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__',
 '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '
__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__seta
ttr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'clear', 'co
py', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update
', 'values']
>>>
```

PROTECO

Conjuntos

Un conjunto es una colección no ordenada de objetos únicos. Con la función integrada dir() podemos conocer sus métodos:

```
>>> dir(set)
['__and__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '_
_eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__ian
d__', '__init__', '__init_subclass__', '__ior__', '__isub__', '__iter__', '__ixo
r__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__', '__rand__',
'__reduce__', '__reduce_ex__', '__repr__', '__ror__', '__rsub__', '__rxor__', '_
_setattr__', '__sizeof__', '__str__', '__sub__', '__subclasshook__', '__xor__',
'add', 'clear', 'copy', 'difference', 'difference_update', 'discard', 'intersect
ion', 'intersection_update', 'isdisjoint', 'issubset', 'issuperset', 'pop', 'rem
ove', 'symmetric_difference', 'symmetric_difference_update', 'union', 'update']
>>>
```

PROTECO

Referencias

2018- Python “The Python Standard Library”. Rcuperado de: https://docs.python.org/3/library/stdtypes.html#numeric-types-int-float-complex

Funciones de listas en Python. Recuperado de : https://likegeeks.com/es/funciones-de-listas-de-python/

PROTECO

Referencias

Tuplas . Rcuperado de: https://uniwebsidad.com/libros/algoritmos-python/capitulo-7/tuplas?from=librosweb

PROTECO