Tipos de Datos

...

Programación
2024

UNRaf UNIVERSIDAD NACIONAL DE RAFAELA

Resolución de problemas (de nuevo)

Análisis	⇨	Entradas y salidas
Diseño	⇨	Pasos y Pseudocódigo
Implementación	⇨	Código
Depuración	⇨	Corrección de errores

Python

- Un lenguaje de programación permite dar **instrucciones** a un **sistema informático**.
- Python es un lenguaje
  - de uso general
  - multiplataforma
  - de (muy) alto nivel.
  - interpretado
  - orientado a objetos
  - de tipado dinámico.
  - con una extensa librería de funciones.
  - con una comunidad activa.

Ejemplo de Programa en Python

```
# Código para calcular una hipotenusa

cat_menor = 3
cat_mayor = 4

hip = (cat_manor ** 2 + cat_mayor ** 2)
hip = hip ** 0.5 # “**” significa “elevado a”

print(hip)
```

Ejemplo de Programa en Python

```
# Código para calcular una hipotenusa

cat_menor = 3
cat_mayor = 4

hip = (cat_manor ** 2 + cat_mayor ** 2)
hip = hip ** 0.5 # “**” significa “elevado a”

print(hip)
```

Ejemplo de Programa en Python

Variables

Operadores

```
# Código para calcular una hipotenusa

cat_menor = 3
cat_mayor = 4

hip = (cat_manor ** 2 + cat_mayor ** 2)
hip = hip ** 0.5 # “**” significa “elevado a”

print(hip)
```

Ejemplo de Programa en Python

```
# Código para calcular una hipotenusa

cat_menor = 3
cat_mayor = 4

hip = (cat_manor ** 2 + cat_mayor ** 2)
hip = hip ** 0.5 # “**” significa “elevado a”

print(hip)
```

Variables

Operadores

Funciones

Ejemplo de Programa en Python

```
# Código para calcular una hipotenusa

cat_menor = 3
cat_mayor = 4

hip = (cat_manor ** 2 + cat_mayor ** 2)
hip = hip ** 0.5 # “**” significa “elevado a”

print(hip)
```

Variables

Operadores

Comentarios

Funciones

Variables

- Permiten almacenar datos en memoria.
- Para asignar un valor a una variable se usa el signo igual (=).
- Si los datos no se guardan en una variable, se pierden luego de ser calculados

```
cat_menor = 3
cat_mayor = 4

(cat_manor ** 2 + cat_mayor ** 2) ** 0.5
# Se hace el cálculo, pero no se tiene acceso
al resultado!!!
```

Variables

- Los nombres de las variables (y de las funciones) deben cumplir ciertas reglas:
  - Empezar con una letra.
  - El resto del nombre puede contener solo letras, números y guiones bajos (_)
  - Mayusculas y minusculas son caracteres distintos (“Variable” y “variable” son dos nombres distintos!)

Palabras reservadas

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

Tipos de Datos

- Primitivos:
  - Enteros (int)
  - Reales (float)
  - Cadenas de caracteres (strings)
  - Verdadero / falso (boolean)
- Abstractos:
  - Tuplas y listas
  - Diccionarios
  - Grafos
  - Arboles
  - ...

} *Algoritmos y Estructuras de Datos!!*

Tipos de Datos Numéricos

- Existen tres tipos de datos numéricos nativos en Python: **enteros**, **flotantes** y **complejos**.
- Los **enteros** pueden ser tanto positivos como negativos:
  - 5
  - -5
- Los **flotantes** son números que admiten decimales:
  - 5.0
  - -1.9
  - 6.022E23
  - 0.0000038
- No existe un límite para el largo de los números (pero para números muy grandes conviene usar notación científica).

Cadenas de caracteres

- Permiten almacenar información en formato de texto.
- Un **string** puede almacenar cualquier tipo de carácter del código ASCII, como letras, números, signos de puntuación y espacios .
- Para definir un string se pueden utilizar tanto comillas simples (‘ ‘) como dobles (“ ”).
    - “Este es un string”
    - ‘Este es otro string’
    - “223344”
    - “.,,,!!”

Codigo ASCII

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

Caracteres especiales comunes

- Para ingresar ciertos elementos en un string es necesario utilizar una sintaxis especial.
- Todos estos caracteres comienzan con una barra invertida (“\”)

Caracter	Funcion
\’	Comilla simple dentro del string
\”	Comilla doble dentro del string
\n	Fin de linea
\t	tabulacion

- El string *I’m learning so much today* se podría almacenar como:
  - string1 = ‘I\’m learning so much today’
  - string2 = “I’m learning so much today”

Tipos de Datos Lógicos

- Los booleanos admiten dos valores: verdadero (**True**) y falso (**False**)
- Son muy utilizados para evaluar alternativas (“si, sino”).
  - **if** (32 > 12) ⟹ True
  - **if** (“arbol” > “bici”) ⟹ False
  - **if** (“vaca” > “ñandu”) ⟹ False

Constantes

- Es un dato que no se modifica en toda la ejecución del programa.
- En Python no existe el tipo de dato “constante”, sino que se definen variables (usualmente, nombres en mayúsculas) que luego no se modifican.
- Algunos ejemplos de constantes muy utilizadas:
    - PI = 3.1415
    - G = 9.81
    - E = 2.7182

Comentarios

- Son líneas en el código que no son leídas por el intérprete.
- Comienzan con un numeral (#).
- Si se desean comentar más de una línea, se pueden utilizar 3 comillas dobles (""")

```
# Esto es una linea con comentarios
var1 = "Esto es una variable del tipo string"
# Esto es otro comentario
var2 = 7 # Los comentarios también pueden ir acá
"""
Acá van
varias
lineas de comentarios
"""
```

Operaciones entre Números

Operaciones aritméticas entre números (int y float):

Operación	Significado
x + y	Suma
x - y	Resta
x * y	Multiplicación
x / y	División
x ** b	Exponencial (“x” elevado a la “y”)
x % y	Módulo (resto de la división)
x // y	División entera

Ejemplos: 7 / 3 = 2.33 7 // 3 = 2 7 % 3 = 1

Operaciones entre Números

Asignaciones entre números (int y float)

Operación	Equivalente a	Significado
x += y	x = x + y	Asigna a “x” la suma entre “x” y “y”
x -= y	x = x - y	Asigna a “x” la resta entre “x” y “y”

Ejemplo:

```
x = 7
y = 5
x += y # ahora “x” vale 12
x -= 2  # ahora “x” vale 10
```

Este tipo de asignaciones funcionan para todas las operaciones entre números

Reglas de Prioridad

- Se respetan las reglas de prioridad de una ecuación matemática tradicional:
  - Primero se evalúa lo que está entre paréntesis.
  - Segundo multiplicaciones, divisiones y módulos.
  - Por último, sumas y restas.
- Ejemplo:

$$3 * (5 - 2) / 7 + 12 - 7 * (4 - 1)$$

Operaciones entre Strings

Operaciones entre strings:

Operación	Significado
a + b	Concatenación
a[i]	Indexación
a[i:j]	Cortar
a * b	Repeticion (b es entero!)

Ejemplos:

```
s1 = “programacion”
s2 = “2023” # Como tiene comillas es un string
s3 = s1 + s2 # Da como resultado programacion2023
inicial = s3[0]
tresletras = s3[2:5] # Devuelve la 3er, 4ta y 5ta letra
```

Comparaciones lógicas

- Dan como resultado un booleano

Operación	Resultado
5 > 9	False
5 < 9	True
5 >= 9	False
5 <= 9	True
5 == 9	False
5 != 9	True

Tipado Dinámico

- En Python, al definir una variable no se debe indicar de qué tipo es.
- Es posible asignarle a una variable un dato de otro tipo.

```
var1 = 7
var1 = False
var1 = “tipado dinamico”
```

Cambiando entre Tipo de Datos

- Para determinar qué tipo de datos tiene una variable, es posible usar la función **type**.

```
>>> var1 = 7
>>> type(var1)
<class 'int'>
>>> var2 = "7"
>>> type(var2)
<class 'str'>
```

- Es posible convertir una variable a otro tipo de datos, siempre y cuando el contenido de la variable sea compatible con ese tipo de dato.

```
>>> var3 = float(var2)
>>> type(var3)
<class 'float'>
>>> var4 = "abc"
>>> float(var4)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: could not convert string to float: 'abc'
```

Corriendo un código

- Existen 2 formas básicas de correr un programa:
  - Modo interactivo
  - Scripting

Corriendo un código: interactivo

- El modo interactivo se basa completamente en una consola.
- El usuario ingresa sentencias de código y obtiene los resultados de manera visual.

```
>>> a = 32
>>> b = 54
>>> a+b
86
>>> type(a)
<class 'int'>
>>>
```

Corriendo un código: *scripting*

- Un ***script*** es un archivo de texto con sentencias que son ejecutadas en orden por un intérprete.
- El ***script*** termina cuando se ejecutaron todas las líneas o cuando se encontró un error.
- No hay posibilidad de corregir errores durante su ejecución

script.py
~/Documents

```
1 var1 = 123.45
2 var2 = 345.67
3 print (var1 / var2)
4 print (var1 * var2)
```

```
matias@matias-laptop:~/Documents$ python script.py
0.3571325252408366
42672.961500000005
```

Errores

- Durante la ejecución de un código, es muy frecuente encontrar errores.
- Los errores se pueden clasificar en 3 tipos:
  - Compilación
  - Ejecución
  - Lógica
- En los dos primeros casos, el intérprete nos dará mensaje con el error que encontró y en qué línea ocurrió.

Errores: compilación

- Son los errores más comunes y están asociados principalmente a errores de sintaxis.

```
>>> 1a = 1234
  File "<stdin>", line 1
    1a = 1234
     ^
SyntaxError: invalid syntax
>>> s1 = "error
  File "<stdin>", line 1
    s1 = "error
               ^
SyntaxError: EOL while scanning string literal
```

Errores: ejecución

- Estos errores ocurren cuando se pide al intérprete realizar una operación que no puede hacer.

```
>>> n1 = 8
>>> n2 = 0
>>> n1 / n2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>> s1 = "errores"
>>> s1[10]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

Errores: lógica

- Surgen si hay algún error en el planteo de la solución.
- Usualmente el programa corre sin problemas, pero el resultado obtenido no tiene relación con lo pedido.

```
>>> cat1 = 3
>>> cat2 = 4
>>> (cat1**2 + cat2**2)**1/2
12.5
>>>
>>> (cat1**2 + cat2**2)**0.5
5.0
```