Argentina Programa 4.0

Universidad Nacional de San Luis

Desarrollador Python

Lenguaje de Programación Python: Tipos de Dato Básicos

Autor:

Dr. Mario Marcelo Berón

UNIDAD 2

LENGUAJE DE PROGRAMACIÓN PYTHON: TIPOS DE DATO BÁSICOS

2.1. Introducción

En esta unidad se comenzará a estudiar en más detalle el lenguaje de programación Python. Se explicarán las reglas que se emplean para asignar un nombre a los objetos que se utilizan en el programa y se mostrarán las palabras claves del lenguaje. Como paso siguiente se explicarán los tipos de datos más importantes que utiliza el lenguaje difiriendo la presentación de los tipos Colección los cuales serán introducidos en la *Unidad 3*. Los tipos de datos que se explicarán en esta unidad se denominan *Tipos de Datos Primitivos* es decir son tipos de datos que ya están incorporados al lenguaje. También se explicarán otros tipos de datos que pertenecen a la librería estándar del lenguaje. La única diferencia entre estos tipos es que para usar los tipos de la librería estándar se debe importar el módulo que lo implementa.

2.2. Identificadores y Palabras Claves

Cuando se crea un dato el mismo se puede asignar a una variable o insertar en una *colección*. Los nombres que se le dan a los objetos que se utilizan en el programa se denominan *identificadores* o *simplemente nombres*. Un identificador válido es una secuencia de caracteres de cualquier longitud que consiste de un carácter de comienzo seguido de cero o más caracteres. Los identificadores deben obedecer las siguientes reglas:

Regla 1: El carácter de comienzo debe ser cualquier carácter *unicode*[1] que se considere una letra incluyendo las letras *ASCII*[2] (" a","b",...,"z",."A","B"-,...,"Z"), el guion bajo "_"también como letras de otros lenguajes no latinoamericanos y anglosajones. Los caracteres que siguen al primero puede ser cualquier carácter que sea permitido como primer carácter y luego caracteres que no sean espacio en blanco incluyendo los caracteres unicode que se consideran números.

Regla 2: Se distingue entre mayúsculas y minúsculas.

Regla 3: No se pueden utilizar identificadores cuyos nombres sean igual al de las palabras claves del lenguaje.

A continuación se muestran las palabras claves que posee el lenguaje Python.

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

[1]Unicode es un estándar de codificación de caracteres diseñado para facilitar el tratamiento informático, transmisión, y visualización de textos de numerosos idiomas y disciplinas técnicas, además de textos clásicos de lenguas muertas.

[2]ASCII es un código de caracteres basado en el alfabeto latino, tal como se usa en inglés moderno. Fue creado en 1963 por el Comité Estadounidense de Estándares como una evolución de los conjuntos de códigos utilizados entonces en telegrafía.

x+y	Suma el número x y el número y
x-y	Resta el número x con el número y
x*y	Multiplica el número x con el número y
x/y	Divide el número x con el número y
x//y	Divide el número x por el número y y trunca la parte fraccionaria
x%y	Produce el resto de dividir x por y
x**y	Eleva x a la potencia y
-x	Si x es distinto de 0 cambia el signo

Cuadro 2.1: Operaciones con Números Enteros

2.3. Tipos Integrales

Python provee dos tipos primitivos integrales: *int* y *bool*. Ambos tipos son inmutables es decir no se pueden cambiar aunque esta característica gracias a la implementación de Python no se nota. El 0 en expresiones booleanas es equivalente a usar *False* y utilizar cualquier valor distinto de 0 es equivalente a usar *True*.

2.3.1. Entero

El tipo entero representa los números enteros. El tamaño de los enteros está limitado solo por la memoria de la máquina, por lo tanto, puede haber enteros de cientos dígitos.

Las operaciones en la tabla 2.1 no son las únicas se pueden encontrar muchas más, varias de ellas tienen que ver con conversiones a otras bases numéricas y operaciones a nivel de bits. Estas operaciones están fuera del alcance de este curso introductorio al lenguaje de programación. Los *literales enteros* se escriben en base 10.

Los objetos de datos de tipo entero se pueden crear utilizando la sentencia de asignación:
x=20
x es una variable de tipo entero que tiene almacenado un 20.

También es posible crear un entero con *int()* en este caso se describen

x and y	Realiza el and lógico entre x e y
x or y	Realiza el or lógico entre x e y
not x	Niega x

Cuadro 2.2: Operaciones con Booleanos

tres formas de uso:

- x=int() inicializa la variable x en 0.
- x=int(3) se invoca con un argumento, en este caso la variable x tiene el valor.
- Si el argumento tiene un tipo diferente se realiza una conversión de tipos. x=int("10",2) crea un entero en base 2.

2.4. Booleanos

El tipo bool representa los valores booleanos True y False.
Como en el caso de los enteros, los booleanos se pueden crear de la siguiente manera:

- b=False crea una variable booleana inicializada en *False*.
- b=bool() crea una variable booleana inicializada en *False*.
- b=bool(True) crea una variable booleana inicializada en *True*.

Es importante mencionar que todo valor distinto de cero o distinto de vacío (en el caso de las estructuras de datos que se verán en las unidades siguientes) se considera *True* y 0 o vacío se considera *False*. Las operaciones básicas que se pueden realizar con los booleanos se muestran en la tabla 2.2. Se hace notar que las operaciones antes mencionadas no son las únicas existen otras que el lector puede encontrar en las bibliografía que describen Python.

x+y	Suma el número x y el número y
x-y	Resta el número x con el número y
x*y	Multiplica el número x con el número y
x/y	Divide el número x con el número y
x//y	Divide el número x por el número y y trunca la parte fraccionaria
x%y	Produce el resto de dividir x por y
x**y	Eleva x a la potencia y
-x	Si x es distinto de 0 cambia el signo

Cuadro 2.3: Operaciones con Números de Punto Flotante

2.5. Tipos de Punto Flotante

Python provee tres clases de números de punto flotante: *flotantes* (float), *complejos* (complex), y el *decimal* (*decimal.Decimal* de la librería estándar).

2.5.1. Float

El tipo float permite representar a los números reales. La siguiente tabla muestra algunas (hay muchas más) operaciones que se pueden realizar con los floats.

El tipo float, al igual que los tipos descritos con anterioridad, tienen un constructor llamado: *float()* el cual se puede invocar de la siguiente manera:

- f=float() crea un número de punto flotante inicializado en 0.0.
- f=float(2.0) crea una copia del argumento y se lo asigna a f. Observe que el argumento es otro flotante.
- f=float("3.2") cuando se invoca con un string como argumento realiza la conversión correspondiente a flotante y se lo asigna a f. En caso de que no se pueda hacer la conversión puede retornar como resultado *NaN* (Not a Number - No es un número) o *infinity* (infinito).

2.5.2. Complejos

Es un tipo inmutable que permite representar un número complejo. Para llevar adelante esta actividad utiliza dos flotantes uno para la parte *real* y otro para la parte *imaginaria*. Los literales complejos se escriben con la parte real y la parte imaginaria unidas por un signo + o - y a la parte imaginaria se le agrega el sufijo j. Ejemplo: 1+2j, 3-12j. Excepto para //, %, *divmod()* y *pow()* con tres argumentos todos los operadores y funciones del tipo entero se pueden utilizar con el tipo *complex*.

2.5.3. Decimal

El tipo decimal permite representar a los números reales inmutables con mucha precisión. Al igual que los tipos previamente descritos, los números decimales se pueden crear con el constructor decimal *Decimal()* se puede utilizar de la siguiente manera:

- b=decimal.Decimal() crea un decimal inicializado en 0.
- b=decimal.Decimal(9876) crea un decimal con la parte entera 9876.
- b=decimal.Decimal("54321.012345678987654321") crea un decimal a partir de un string.

Es importante notar que en el caso de que el constructor reciba argumentos estos deben ser *int* o *string* no pueden ser *float* porque no tienen la precisión suficiente para representar números reales. Otro aspecto que se tiene que tener en cuenta es que se debe *importar la librería decimal* para poder usar el tipo. Se puede convertir un *float* a un *decimal* utilizando la función *decimal.Decimal.from_float()*. Esta función toma como argumento un *float* y retorna como resultado un *decimal.Decimal*. Todas las operaciones que se utilizan para enteros (int y float) también se pueden usar para decimales (decimal).

2.5.4. Strings

Las cadenas de caracteres (strings) son representados por el tipo *str* el cual mantiene una secuencia de caracteres *unicode*. El constructor del tipo (str) se puede invocar de diferentes maneras:

- s=str() cuando se invoca sin argumentos retorna como resultado un string vacío, el cual en este caso se asigna a s.
- s=str("Hola Mundo") cuando se invoca con un argumento de tipo str retorna como resultado una copia del string la cual, en este caso, se asigna a s.

str() también se puede utilizar para realizar conversiones de tipo. En este caso el argumento es un *string* o un *valor que se pueda convertir a un string*. Por ejemplo: s= str(20) en este caso 20 que es *int* se convierte a *str* y esa copia se asigna a s. Los literales se pueden crear utilizando comillas simples, dobles o triples. Las comillas triples permiten generar un string multilínea. Por ejemplo: s="Hola Mundo" o s='Hola Mundo' o s=" " " Hola Mundo " " ".

Todas esas asignaciones hacen que s apunte al string "Hola Mundo". El uso de diferentes tipos de comillas es útil cuando se desea que un string contenga en su interior comillas sean simples o dobles. Por ejemplo: s= "Presione 'ENTER' para continuar".

Operaciones con Strings

Comparaciones: Los string soportan los operadores relacionales usuales a saber: <, <=, ==, !=, >, and >=.

Rebanadas: Para extraer un carácter de un string es posible utilizar el operador []. No obstante, para este tipo este operador es mucho más versátil y puede ser utilizado no solo para extraer un carácter sino una secuencia de caracteres.

- Extracción de Caracteres Individuales desde un String: Comúnmente esta operación se realiza utilizando índices que van desde

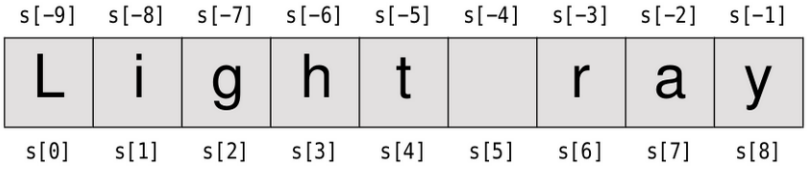


Figura 2.1: Índices Negativos

0 hasta la longitud del string menos 1. Sin embargo, en este caso también es posible usar índices negativos ver figura 2.1.

Los índices negativos son muy útiles dado que permiten acceder a los caracteres de forma más sencilla. Por ejemplo el índice -1 se corresponde con el último elemento del string. En el caso de que se coloque un índice fuera de rango se disparará una excepción *IndexError*. El operador *rebanada* (slice) tiene tres sintaxis:

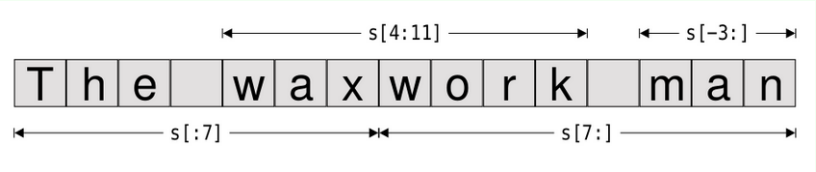
- sec[comienzo]: retorna como resultado el caracter del string sec que se encuentra en la posición comienzo.
- sec[comienzo:final]: retorna como resultado la secuencia de caracteres de sec comprendidos entre comienzo y final menos 1.
- sec[comienzo:final:paso]: retorna como resultado la secuencia de caracteres de sec comprendidos entre comienzo y final saltando una cantidad indicada por paso.

Comentario Importante

- Es importante tener presente que *comienzo*, *final* y *paso* son índices de tipo entero.
- Otro aspecto a tener en cuenta es que estas operaciones no solo son válidas para el tipo string o sino también, como se verá más adelante, para *listas* y *tuplas*.
- Si se usa la segunda opción (un solo :) se puede omitir los índices enteros. Si se omite el índice comienzo se asume que es 0. Si se omite el índice final se asume que es la longitud de la secuencia (*len(sec)*). Si se omiten ambos índices es equivalente a colocar *sec[0:len(sec)]*.

Ejemplos de Rebanadas

A continuación se muestran algunas formas de obtener una rebanada teniendo en cuenta la asignación: s = "The waxwork man".

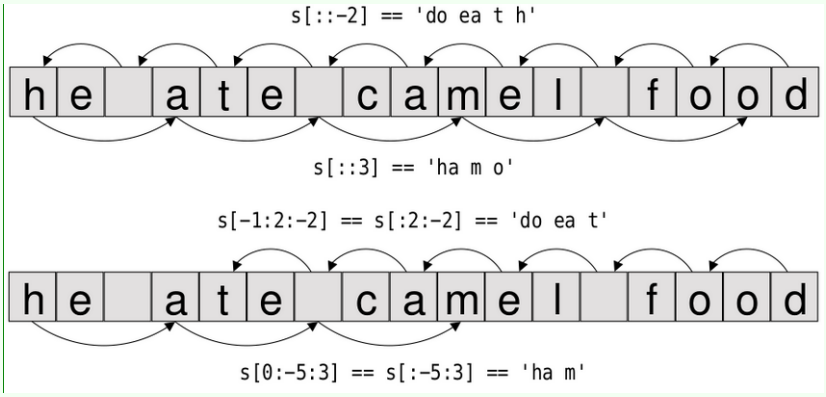


Como fue mencionado con anterioridad la tercera sintaxis de rebanadas, aquella con dos :, no extrae todos los caracteres sino aquellos indicados por el paso. En el caso si se omite el primer índice se asume que es 0 a no ser que el paso sea negativo con

lo cual se asumirá que el índice es -1. Cuando el índice que se omite es el segundo se asumirá que es el *len(sec)* a no ser que el paso sea negativo, en ese caso el índice será 0. Si se omite el paso se asume que será 1 (este uso es no tiene sentido dado que es el comportamiento por defecto de la sintaxis con un único ☻).

Ejemplos de Rebanadas

A continuación se muestran algunos ejemplos de las ideas descritas con anterioridad.



Operadores y Métodos de String: Los string son objetos con tamaño, por consiguiente se puede llamar a *len()* con un string como argumento de la función el resultado será la cantidad de caracteres en el string (cero para el string vacío). El operador + está sobrecargado y cuando se utiliza con strings produce la concatenación de las cadenas que recibe como parámetro. El operador * permite replicar un string así s= " = "* 5 almacena la cadena "=====" en s. El operador de membresía *in* retorna *True* si el string de la izquierda es un substring del string de la derecha. Si se realiza la siguiente actividad *"Hola" in "Hola Mundo"* el resultado será *True* dado que " Hola " es un substring de "Hola Mundo". En otro caso el operador retronará como resultado *False*. En los casos donde se desea encontrar la posición de un string dentro de otro string se pueden usar dos métodos: *str.index()* y *str.find()*. Ambos métodos aceptan como parámetro un string y dos argumentos opciona-

les adicionales que indican la posición de comienzo y la posición final de la búsqueda. El primer método *str.index()* retorna como resultado la posición donde comienza el substring o una excepción *ValueError* si dicho substring no se encuentra en el string de búsqueda. El segundo método *str.find()* retorna como resultado la posición donde comienza el substring o -1 cuando el mismo no se encuentra en la cadena destino. Es importante mencionar que el tipo string tiene muchas funcionalidades que le facilitan la tarea de programar al programador. No obstante, hacerlas explícitas hacen al documento muy denso. Por tal motivo, en $https://www.youtube.com/watch?v=CSGedJV6Yv8\&t=1s$ el interesado puede encontrar explicaciones otras funciones interesantes de string de Python.

Formato de String con el Método str.format(): El método str.format() provee una forma muy flexible y poderosa de crear strings. Este método retorna un nuevo string con los campos de reemplazo con sus argumentos adecuadamente reemplazados.

Sustitución de Parámetros

" El premio nobel 0 fue publicado en 1".format("Hard Times",1854)
" El premio noble Hard Times fue publicado en 1854"

El campo de reemplazo se identifica por un nombre de campo encerrado entre llaves. Si el nombre es un entero se interpreta como una posición en los argumentos pasados al método. De esta manera 0 se corresponde con el primer argumento, 1 se corresponde con el segundo y así siguiendo. En el caso en que se deseen incluir llaves en string de salida las mismas se deben duplicar.

Colocar Llaves en el String de Salida

"{{{0}}} {1} ;-}}".format(" Estoy entre llaves", "Yo no")
"{Estoy entre llaves} Yo no ;-}"

Comentario Importante

El método str.format() es muy completo en esta sección solo se ha comentado la forma más sencilla del mismo. A medida que se avance con el curso y se necesiten nuevas funcionalidades de format se mostrarán versiones más avanzadas del método.

2.6. Ejercicios

Ejercicio 1: Escriba un programa que permita que el usuario ingrese un número entero y luego lo imprima por pantalla.

Ejercicio 2: Escriba un programa que permita que el usuario ingrese un número flotante y luego lo imprima por pantalla.

Ejercicio 3: Escriba un programa que permita que el usuario ingrese dos números e imprima por pantalla la suma, resta y multiplicación de dichos números.

Ejercicio 4: Escriba un programa que permita calcular el área y perímetro de un triángulo.

Ejercicio 5: Escriba un programa que permita calcular el área y perímetro de un cuadrado.

Ejercicio 6: Escriba un programa que permita calcular el promedio de cinco números ingresados por el usuario.

Ejercicio 7: Escriba un programa que imprima por pantalla “Hola Mundo!”

Ejercicio 8: Escriba un programa que solicite al usuario un string y luego imprima por pantalla dicho string.

Ejercicio 9: Escriba un programa que permita almacenar un string en la variable s y luego muestre por pantalla el contenido de s.

Ejercicio 10: Escriba un programa que permita que el usuario ingrese el nombre de una persona y luego imprima por pantalla "Hola:" seguido del nombre de la persona.

Ejercicio 11: Escriba un programa que permita ingresar por teclado el nombre de una persona, la cantidad de horas trabajadas y luego el costo de la hora. El programa debe informar cuánto debe cobrar la persona.

Ejercicio 12: Escriba un programa que calcule el IMC (índice de Masa Corporal).

Ejercicio 13: Escriba un programa que pida al usuario dos números m y n el programa debe imprimir por pantalla el cociente y el resto de la división de m por n. Para este ejercicio asuma que n no puede ser 0.

Ejercicio 14: Escriba un programa que permita realizar la conversión a dólares y euros de una cantidad de pesos ingresada por el usuario.

Ejercicio 15: Escribir un programa que pregunte al usuario una cantidad a invertir, el interés anual y el número de años, y muestre por pantalla el capital obtenido en la inversión.

Ejercicio 16: Una juguetería tiene mucho éxito en dos de sus productos: payasos y muñecas. Suele hacer venta por correo y la empresa de logística les cobra por el peso de cada paquete así que deben calcular el peso de los payasos y muñecas que saldrán en cada paquete a demanda. Cada payaso pesa 112 g y cada muñeca 75 g. Escriba un programa que lea el número de payasos y muñecas vendidos en el último pedido y calcule el peso total del paquete que será enviado.

Ejercicio 17: Imagine que acaba de abrir una nueva cuenta de ahorros que ofrece el 4 % de interés al año. Estos ahorros debido a intereses, que no se cobran hasta finales de año, se añaden al balance final de su cuenta de ahorros. Escriba un programa que comience leyendo la cantidad de dinero depositada en la cuenta de ahorros, introducida por el usuario.

Después el programa debe calcular y mostrar por pantalla la cantidad de ahorros tras el primer, segundo y tercer años. Redondear cada cantidad a dos decimales.

Ejercicio 18: Escriba un programa que permita ingresar una cantidad de dinero c y un porcentaje p. El programa debe calcular el porcentaje p de dinero de c.