


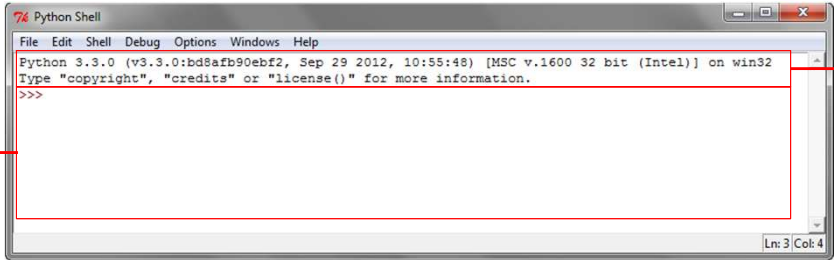
UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
Fundamentos de Computación y Programación (10110-1)



CLASE N°3

Python como calculadora

Entorno Python



Información del intérprete

Espacio para ingresar
instrucciones al intérprete,
como **operaciones
matemáticas**

2

Operadores




- Operadores aritméticos
 - Suma +
 - Resta -
 - Multiplicación *
 - División /
 - Exponenciación **
 - Módulo o resto %



3

Valores numéricos



- Python maneja **diferentes tipos de números** y entrega los resultados dependiendo de qué clase de números se esté utilizando:
 - Enteros (**int**): Para enteros en el intervalo [-2147483648, 2147483647]
 - Enteros largos (**long**): Para enteros fuera del rango de los enteros de 4 bytes (Ej: 50394034940032L)
 - Flotantes o **float**: Para números no enteros, **no son iguales a los números reales** (Ej: 2.54)

4

Valores numéricos



- Python prefiere **mantener el tipo de dato**
 - Vimos, por ejemplo, que si se operan **dos enteros**, entonces Python intentará devolver un resultado **entero**
 - Pero si se **combinan** tipos numéricos en una expresión aritmética, Python **generaliza** todos los valores antes de operarlos
 - Enteros largos son más generales que los enteros
 - Los **flotantes son más generales** que los enteros largos y los enteros

5


Precedencia



Operación	Operador	Aridad	Asociatividad	Precedencia
Exponenciación	**	Binaria	Derecha	1
Identidad	+	Unaria	—	2
Cambio de signo	-	Unaria	—	2
Multiplicación	*	Binaria	Izquierda	3
División	/	Binaria	Izquierda	3
Módulo	%	Binaria	Izquierda	3
Suma	+	Binaria	Izquierda	4
Resta	-	Binaria	Izquierda	4

6

Precedencia




- Python sigue estas reglas de precedencia:

$5 + 5 ** 3 / 5 * 4 - 10 * 3$

The diagram illustrates the evaluation order of the expression $5 + 5 ** 3 / 5 * 4 - 10 * 3$ using numbered brackets (1-6) to show the sequence of operations from left to right, following operator precedence. The final result, 75, is shown in a box at the end of the sequence.

7

Precedencia




- Es decir, la evaluación corresponde a la siguiente expresión con paréntesis:

$5 + ((5 ** 3 / 5) * 4) - 10 * 3$

The diagram illustrates the evaluation order of the expression $5 + ((5 ** 3 / 5) * 4) - 10 * 3$ using numbered brackets (1-6) to show the sequence of operations from left to right, following operator precedence. The final result, 75, is shown at the end of the sequence.

8

Variables



- Una **asignación** es una sentencia con la siguiente estructura:

<Identificador> = <expresión>


↑

¡Recuerden que no es una comparación ni una equivalencia matemática!

- Con la asignación se definen **variables** y **constantes** para almacenar valores

9

Variables



- Una **asignación** es una sentencia con la siguiente estructura:

<Identificador> = <expresión>

- Reglas de un identificador:
 - El primer carácter no puede ser un dígito
 - Puede llevar letras, dígitos y el carácter subrayado (_)
 - No puede coincidir con palabras reservadas:
 - and, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, yield*

- Puede ser
 - un valor constante (un número o texto)
 - una operación entre números
 - una operación entre variables y constantes previamente declaradas
 - Mezcla entre operaciones, variables y números

10

Variables



- Para ver el contenido de una variable o constante, **definida previamente**, podemos usar el comando **print**

```
>>> print <expresión>
```

- Si la expresión es un texto, se pueden mostrar mensajes en pantalla:

```
>>> print "mensaje"
```

- También se pueden mezclar con más de un argumento:

```
>>> print "mensaje 1", <expresión 1>, <expresión 2>, "mensaje 2"
```

11

PARA LA PRÓXIMA CLASE



- Revisar guía de ejercicios propuestos:
 - Profundizar en los contenidos vistos hoy
- Se utilizarán **variables** y **constantes** para otro tipo de valores numéricos y trabajaremos en Python con expresiones más complejas que incluyen **funciones**

12

