PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC1103- INTRODUCCIÓN A LA PROGRAMACIÓN

Laboratorio 1
Introducción a Python: tipos, operaciones y variables

Segundo semestre, 2015

Objetivo

Este laboratorio inicial es para que te familiarices en el ambiente de Python. El objetivo principal es explorar los tipos de datos, las operaciones que se pueden realizar con ellos, y cómo asignar variables e incorporarlas en las operaciones. Veremos casos de sintaxis incorrecta para que ganes conocimiento en el manejo de errores. Tambien veremos el uso de las funciones básicas como `type()`, `print()` e `input()`, que te serán muy útiles a lo largo del curso.

Se asume que sabes iniciar una consola usando Python. Si no es el caso, el ayudante te asistirá.

1. Primeros pasos con Python

Iniciar Python depende del sistema en que te encuentres (Windows, Linux, Mac OSX). El nombre del ejecutable de Python también puede variar (puede ser `python`, `python3.2`, etc.); en nuestros ejemplos, se asume que el programa ya está iniciado. En este laboratorio los ejemplos se asumen que se correrán el modo Consola (”Shell”), aunque puedes usar el que más te acomode (IDLE, IPython, etc.).

Indistintamente del modo en que lo inicies, el siguiente mensaje aparecerá en la consola:

```
Python 3.2.3 (default, Jun 16 2015, 14:46:58)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

En el ejemplo, `>>>` nos indica que podemos escribir los comandos (la llamada *linea de comandos* de Python). Por ejemplo, si escribes un `2`, y presionas `<Enter>`, éste se replicará:

```
>>> 2
2
```

En este ejemplo, el número replicado es la salida del comando. Prueba con letras o texto; te indicará que cometes un error, lo que te quedará claro más adelante.

Probemos algo más complejo. Veremos la primera función: `print()`. Escribe en la línea de comandos `print("hola mundo")`, y presiona `<Enter>`

```
>>> print("hola mundo")
hola mundo
```

En este segundo ejemplo, le entregamos `"hola mundo"` a la función `print()`, y ésta despliega el mensaje como salida. Prueba a poner número enteros, decimales, o frases más largas dentro de la función `print()` para que te familiarices con ella.

2. Tipos de datos y operaciones

Python tiene varios tipos (clases) de datos básicos, que se muestran en la siguiente tabla:

Tipo	Clase	Ejemplo
Números enteros	`int`	`2`
Números reales	`float`	`2.5`
Números complejos	`complex`	`2 + 3j`
Valores booleanos	`bool`	`True`/`False`
Cadenas de texto	`str`	`"hola mundo"`

Tabla 1: Tipos de datos básicos

Es importante notar algunos aspectos de los tipos básicos:

- El símbolo decimal de números reales es el punto (`.`). No se usa coma para definir números.
- En números complejos, es lo mismo escribir `2+4j` o `4J+2`, no importa el orden o si es `j` o `J`.
- Las cadenas de texto siempre deben estar entre comillas dobles (`"hola mundo"`) o entre apóstrofes (`'hola mundo'`). Se debe ser consistente en usar uno u otro en la misma cadena.
- Valores booleanos siempre comienzan en mayúsculas, `True` o `False` está correcto.

La función `type()` nos servirá para saber el tipo de valor de una expresión. Pruebe con lo siguiente en la línea de comandos:

```
>>> type(1.2)
>>> type(100)
>>> type("hola de nuevo")
>>> type(false) # notar que está erroneamente escrito
```

Es importante notar que "Clase" en la Tabla 1 (`int`, `float`, etc.) define qué tipo de *operaciones* podemos hacer entre dos tipos. Por ejemplo, realice las siguientes expresiones y vea la salida de la línea de comandos.

```
>>> 20 + 3
>>> 45.7/2
>>> 2+3j + 5
>>> (2+3j) + 5
>>> True*False
>>> False + True
>>> "esta cadena " + "quedara pegada"
```

Como puedes observar, la mayoría de las operaciones aritméticas entre valores numéricos (e incluso booleanos) son expresiones válidas. Como viste en el último ejemplo, la operacion de suma entre dos cadenas de texto las concatena, es decir, une el final de la primera con el inicio de la segunda.

Otro aspecto inportante es que las clases de valores (como `int`, `str`, etc.) son además funciones que convierten su argumento en el tipo de valor que indica la función. Pruebe con lo siguiente en la línea de comandos:

```
>>> int("5")
5
>>> float("3.14159")
3.14159
>>> str(5.5)
'5.5'
```

En Python, están definidas las operaciones aritméticas de la siguiente tabla. Te recomendamos hacer los ejemplos en la línea de comandos de Python (algunos pueden dar error, lee los mensajes para saber el problema):

Operación	Descripción	Ejemplo 1	Ejemplo 2	Ejemplo 3
`+`	Suma	`2.3+5.4`	`1+0.1`	`"hola"+ 5`
`-`	Resta	`45.45-10.02`	`3j-2j`	`True - False`
`-`	Negación	`-5.4`		
`*`	Multiplicación	`(2.3+4.2j)*3`	`100000*100000`	`"hola"*5`
`**`	Potenciación	`2**8`	`2.71 ** (-3.159j)`	`0 ** 0`
`/`	División	`100/99`	`(1+1j)/(1-1j)`	`1/0`
`//`	División entera	`100//99`	`100//50`	`100//2.5`
`%`	Módulo	`10%3`	`10%10`	`10%15`

Tabla 2: Operadores aritméticos

La jerarquía de ejecución de los operadores aritméticos es: paréntesis, potenciación, multiplicación/división, suma/resta. También en todas las operaciones es posible combinar dos o más valores, por ejemplo en `2+3+4`, `3-2/3` o `2*3**3`. Siempre ten presente la jerarquía de cálculo, ya que por ejemplo `3-2/3` no es lo mismo que `(3-2)/3`. Cuando quieras que un cálculo se efectúe antes que el resto, asegúrate de encerrarlo en paréntesis.

Los operadores aritméticos operan sobre valores numéricos (y algunos sobre cadenas de texto), entregando un resultado que es numérico o `str` según sea el caso. Existen también los llamados *operadores lógicos o booleanos*, como las comparaciones para valores numéricos (==, !=, >, <, >=, <=) y para valores booleanos (`and`, `or` y `not`. Todos los operadores lógicos entregan un resultado que es un valor booleano (`True` o `False`). Es muy importante notar desde ya que el operador de comparación es == y NO =. La Tabla 2 resume los operadores lógicos, haz los ejemplos para que te familiarices con su uso.

Operación	Descripción	Ejemplo 1	Ejemplo 2
`== (!=)`	Igual (distinto) a	`2 == 2`	`True != False`
`<(<=)` `>(>=)`	Menor (menor o igual) Mayor (mayor o igual)	`1<1.1` `True >= False`	`2.3 <= 1000+1j` `"b">"a"`
`and`	Ambos `True`	`True and True`	`False and True`
`or`	Alguno `True` o `False`	`False or 2==2`	`1>1 or 1<1`
`not`	Negación	`not True`	`True or not True`

Tabla 3: Operadores booleanos

Es interesante notar que comparaciones entre cadenas de texto son llevadas letra por letra, siguiendo el orden del alfabeto. Siempre las minúsculas son "mayores" que las mayúsculas.

Hasta ahora, vimos que `print("hola mundo")` escribe `hola mundo` como salida. Podemos poner cualquier tipo de dato dentro de print, e incluso expresiones con operaciones, y el resultado será impreso en pantalla. Además, `print()` acepta dos o más argimentos separados por comas, y los imprime uno tras otro separdos por un espacio. Practica con los suguientes ejemplos escribiéndolos en la línea de comandos, donde la salida de todos es `"Me sacare un 7"`:

```
>>> print("Me sacare un 7")
>>> print("Me sacare un ","7")
>>> print("Me sacare un", " ", "7")
>>> print("Me sacare un", 7)
>>> print("Me","sacare","un", 2+5)
>>> print("Me","sacare","un", str(2*3+1))
```

3. Variables

Hasta ahora hemos aprendido a usar la línea de comandos de Python como una calculadora más o menos estándar. Todo se pone más interesante cuando comenzamos a usar *variables*, que es un nombre que asignas para usarlo como un *alias* del valor que representa. En Python, la asignación de variables se realiza con el símbolo de igualdad (=). Para comenzar, haremos una asignación del valor `1` a una nueva variable `x`. Escribe lo siguiente en la línea de comandos de Python:

```
>>> x = 1
```

A diferencia de cuando escribíamos una operacion o un número, no hay ningun resultado. Sin embargo, si escribimos `x` y presionamos `<Enter>`, nos mostrará su valor:

```
>>> x
1
```

Tal como en el primer ejemplo de este laboratorio (donde escribíamos un `2` y se replicaba), al escribir `x` se replica su *valor*. Se puese pensar como que `x` *representa* al valor `1`, por lo que se puede usar indistintamente en funciones `type(x)`, `print(x)`, usarla en operaciones como `x+1` o `x>0`, entre todas las operaciones posibles que permita su clase de valor. Pruebe las siguientes asignaciones e imprima (`print(x)`) el valor que toma `x` cada vez que la asigne:

```
>>> x = 2
>>> x = x + 4
>>> x = 2**x - 32
```

Al imprimir los valores (partiendo de `x=1`), `x` toma los valores 2, 6 y 32. Es importante que note que en la asignación de variables, el literal de la izquierda toma el valor del resultado de la expresión de la derecha del signo igual, *no es una ecuación!*. Pruebe con asignaciones de variables de cadenas de texto, como `texto = "hola"`, y realice algunas operaciones, por ejemplo `texto + "de nuevo"` o `texto=="otrotexto"`.

Los nombres de variables pueden tener cualquier largo mientras no tengan espacios ni comiencen por un número (un nombre como `7up` como variable no es permitido!). Es importante notar que existen un mínimo de requerimientos para asignar los nombres de variables:

- Siempre deben comenzar por una letra.
- El único caracter alfanumérico pemitido es el `"_"`, que te servirá para separar nombres, por ejemplo en la asignacion `velocidad_del_sonido=343`.
- No deben tener espacios.
- No deben ser alguna de las 31 palabras reservadas de Python, como: 'False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield'. Cualquier otra palabra es válida como variable.
- Notar que Python es sensible a mayúsculas y minúsculas, por lo que `x` y `X` xon variables distintas.

En muchos programas se necesita la interacción con un usuario. Para esto, Python provee de la función `input()`, la que usamos cuando queremos que el usuario escriba el valor de la variable en modo interactivo. Por ejemplo, el siguiente comando mostrará el mensaje `Ingrese nota Tareas: `, esperando que el usuario ingrese un número (en el ejemplo, ingresamos `6.5`), y luego presione `<Enter>`. El valor que ingresó se almacena en la variable PT con tipo de cadena de texto `str`. Para convertirlo a un número, se utiliza la función `float()`.

```
>>> PT = input("Ingrese nota Tareas: ")
Ingrese nota Tareas: 6.5
>>> print(PT)
```

```
6.5
>>> type(PT)
<class 'str'>
>>> fPT = float(PT)
>>> print(fPT)
6.5
>>> type(fPT)
<class 'float'>
```

4. Ejercicios

Es bueno familiarizarse con el manejo de variables, expresiones y operadores, ya que es la base para construir estructuras más complejas. Realice los siguientes ejercicios pensando en cómo expresarlos mediante el uso de lo que practicamos hoy.

1. La suma de los primeros n numeros naturales está dada por $n(n+1)/2$. Calcule la suma de los primeros 10, 100 y 1000000 números naturales.
2. Usando el operador `%`, y el operador de igualdad ==, indique mediante un resultado `True` o `False` si un número almaceado en una variable `x` es par o impar.
3. Calcule el volumen de una esfera, dada por $\frac{4}{3}\pi r^3$, definiendo $\pi = 3{,}1416$. Calcúlelo para radios de 1, 100 y 6371.2.
4. Encuentre el valor de la expresión $-e^{-\pi j}$, usando $e = 2.718281828459045$ y $\pi = 3.141592653589793$. (Ayuda: j puede escribirse como $0 + 1j$)
5. Encuentre el error (si hay) en las siguientes expresiones:

   a) `>>> 1/0`

   b) `>>> 2*1***3`

   c) `>>> print(texto)`

   d) `>>> texto="aquisedefinebien"`

   e) `>>> print(texto)`

   f) `>>> 11 // 2 + 1j`

   g) `>>> 11 // (2 + 1j)`

   h) `>>> 11 / (2 + 1j)`

6. Encuentre 6 formas de generar el mensaje *Espero terminar antes de las 18 : 00* en pantalla, como por ejemplo

```
>>> print("Espero terminar antes de las",18,": 00")
```

7. Puedes ejecutar un programa usando un archivo de texto, por ejemplo escribiendo tus comandos en un archivo `notas.py`, para luego ejecutarlos desde una consola (Shell) mediante el comando `python notas.py`. Realice un programa para preguntar al usuario cuatro notas (PT = Tareas, PI = Interrogaciones, NE = Examen y PP = Presentación), para luego imprimir en pantalla la nota final del curso $0.3PT + 0.3PI + 0.3NE + 0.1PP$.