Resumen Control 1

Profesor: Juan Alvarez **Auxiliares:** Pablo Cano – Willy Maikowski

Variables

Representación simbólica de un valor almacenada en la memoria del computador. Esta debe ser inicializada con un nombre y comúnmente puede cambiar su valor. Hay varios **tipos**:
(int) Contiene un numero entero (-5, 0, 12)
(float) Contiene un numero real (3.14, 6.0)
(String) Contiene una cadena de texto (“hola”)
(boolean) Contiene expresión lógica (True)

Para **asignar** un valor a una variable:

(Sintaxis) variable=expresión
(Semántica) Primero evalúa la expresión y ese resultado se lo asigna a la variable.
Ejemplo: s = (10+5)*2 Según la semántica primero se lee la expresión, cuyo resultado es 30. El cual luego se asigna a la variable de nombre s.

Operaciones para variables numéricas:

Orden	Tipo	Signos	Ejemplo
1	Unarios	+, -	-1
2	Potencia	**	2**3
3	Multiplicativo	*, /, %	2*4
4	Aditivos	+, -	1+1

Cuidado especial con división de números enteros, resultados dará numero entero.
(Ejemplo) 1 / 2 = 0
(Solución) 1.0 * (1 / 2) = 0.5

Funciones

Grupo de instrucciones que se ejecutan al llamarlas por el nombre asociado. Siempre retornan algo.

Existen algunas **predefinidas** en python

print Lo que hace es imprimir en pantalla lo que se ponga a continuación de la llamada de la función. Si se desea imprimir varios elementos deben ser separados por una coma.
Ejemplo: print “hola”, 4, “cuatro”
input(...) Espera a que el usuario mediante pantalla entregue un valor. Este valor puede no perderse y ser asignado a una variable. Dentro de los paréntesis puede ponerse una cadena de texto (comúnmente que indique lo que se pide).
Ejemplo: nombre = input(“ingrese su nombre”)
abs(...) Entrega el valor absoluto del numero dentro de los paréntesis.
Ejemplo: abs(-1) = 1
max(...) o min(...) Entrega el valor máximo o mínimo, según corresponda, de todos los valores puestos dentro de los paréntesis (separados por coma)
Ejemplo: max(1, 2, 8, -3) = 8

En ocasiones para poder llamar algunos tipos de funciones hay que importar archivos. Para eso se utiliza import
Ejemplo: import math

Funciones **matemáticas** (import math)

Función	Significado	Ejemplo	R
math.sqrt(x)	Raiz de x	math.sqrt(4)	2.0
math.pow(x,y)	x**y	math.pow(4, 2)	16.0
math.exp(x)	e**x	math.exp(1)	2.71
math.log(x)	ln(x)	math.log(e)	1.0
math.sin(x)	Seno de x	math.sin(pi)	0.0
math.cos(x)	Coseno	math.cos(pi)	-1.0
math.tan(x)	Tangente	math.tan(pi)	0.0
math.asin(x)	Arco-seno	math.asin(0)	3.14
math.acos(x)	Arco-cos	math.acos(-1)	3.14
math.atan(x)	Arco-tang	math.atan(0)	3.14

*Pi y e se deben representar como math.pi y math.e

Resumen Control 1

Profesor: Juan Alvarez **Auxiliares:** Pablo Cano – Willy Maikowski

Funciones para números **azar** (import random)

Funcion	**Significado**
random.random()	Real al azar en el intervalo [0,1[
random.randint(x,y)	Entero al azar en el intervalo [x, y]

El programador también puede crear sus propias funciones siguiendo una sintaxis definida. Estas podran ser llamadas dentro del mismo programa.

(Sintaxis)

```
def nombrefuncion(parametros):
        instrucciones
        return expresion
#Fijarse en la indentación (sangría) de las
#instrucciones, la palabra clave def, los dos
puntos #y el retornar algo con return
```

Ejemplo:

```
def perimetroTriangulo(x, y, z):
        return x+y+z
```

If, elif y else

¿Como podríamos hacer si quisiéramos ejecutar ciertas instrucciones solo en el caso de que se cumpla una condición? Se tienen los métodos if, elif y else.
Pero antes de eso debemos saber escribir las condiciones mediante los operadores respectivos.

Operadores **Lógicos**

Orden	Operador	Operación
1	==	Igual que
1	!=	Distinto que
1	<, <=	Menor (o igual) que
1	>, >=	Mayor (o igual) que
2	not	Negación
3	and	Conjunción
4	or	Disyunción

If, elif and else

(Sintaxis)

```
if condicion1:
        instrucciones1
elif condicion2:
        instrucciones2
else:
        instrucciones3
#Fijarse en la ¡indentacion!
```

Esto quiere decir que, si se cumple la condicion1, osea si es verdadera, ejecuta las instrucciones1, si no, si no se cumple esa condicion1 pero si se cumple la condicion2, se ejecutan las instrucciones2. Finalmente si ninguna de las condiciones es verdadera, se ejecutan las instrucciones3.

Ejemplo:

```
if x==y:
        print “x es igual a y”
elif x>y:
        print “x es mayor a y”
else:
        print “x es menor a y”
```

While

Similar al if, hay ocasiones en que uno desea realizar algunas instrucciones repetitivamente mientras se cumpla una condición. Para ello tenemos el ciclo While y una de sus claves es leer su significado en español.

(Sintaxis)

```
while condición:
        instrucciones
#¡¡¡Indentación!!!
```

Ejemplo:

```
n=1
while n<=10:
        print n
        n=n+1
```

Como se dijo anteriormente la clave esta en saber

leer su significado. En este caso si lo leemos nos dice que: *"ene es igual a 1. Mientras el ene sea menor o igual que diez, ejecutaremos las instrucciones que son imprimir ene y aumentar en uno el valor de ene"*
Esto nos ayuda porque hay que fijarse siempre de dos cosas: **Comprobar que siempre termina** el ciclo (en el caso anterior el n siempre va aumentando por lo que va a llegar algún momento en que sea mayor a 10 provocando el termino del ciclo) y fijarse bien que instrucciones queremos que se ejecuten repetitivamente y cuales no.
Otro tema que podría ayudar es siempre actualizar la variable que es utilizada en la condición dentro de las instrucciones del ciclo.

Recursividad

Finalmente, y totalmente equivalente a un while, podemos formar un ciclo llamando a una función dentro de la misma función. A esta técnica se le conoce como recursividad.

(Sintaxis)

```
def función(parámetros):
        instrucciones, condiciones...
        llamada a función(nuevos parámetros)
#¿Les había dicho sobre la indentación?
```

Ejemplo:

```
def factorial(x):
        if x==0:
                return 1
        else:
                return x*factorial(x-1)
```

Una forma de saber si uno puede ocupar recursividad es fijarse en los parámetros que uno puede entregar a la función ¿Como? Bueno, uno debe estar seguro que puede darle la totalidad de información a la recursividad de manera que en algún momento se termine (Al igual que el while). Un ejemplo de esto es la función potencia10(x). Se esta buscando que esta calcule el numero dentro del paréntesis elevado a 10. Fíjense que la forma de llamar a la función es mediante un solo parámetro (el numero que queremos elevar a 10), pero no tenemos como pasarle a la siguiente ejecución de la función la información de en que llamada vamos, por lo que a priori no podemos informarle cuando terminar la recursividad. Por lo que no podríamos ocuparla.