APRENDER A PROGRAMAR CON ALGORITMOS Y PYTHON

por Jair Gaxiola

SESSION 1:

¿Porque quieres programar?

¿Que se necesita para programar?

- **Paciencia, y mucha:** La programación puede llegar a ser muy frustrante y en muchas ocasiones podemos quedarnos pegados por un buen rato en la busqueda de una solución a un problema, por muy pequeño que sea.
- **Perseverancia:** En la programación siempre hay que ser perseverante para poder lograr los objetivos que nos proponemos.
- **Mirar las situaciones desde distintos puntos:** Cuando se nos presenta una situación de cualquier tipo, mientras de más puntos o lados la miremos y obtengamos ciertas conclusiones, aumentamos más la posibilidad de encontrar una buena solución.
- **Pensar lógica y básicamente:** Cuando necesitamos encontrar una solución, debemos enfrentar la situación como un computador lo haría sin saber otro tipo de informaciones que nosotros sabemos y debemos diseñar nuestras soluciones de la forma más básica para poder implementarlas.
- **Ser estructurado:** Bueno, para hacer esto, primero hay que hacer esto, luego esto, luego esto y finalmente esto. Ese orden y esa estructuración nos irá ayudando a no tener que volver sobre el mismo camino hecho para agregar cosas que deberían haber estado ahí.
- **Conocimientos matemáticos**
- **Ser curioso y tener disposición a resolver problemas:** Por eso es bueno tener curiosidad de como resolverlos de distintas formas y siempre tener la disposición a encontrar soluciones, en especial la más adecuada.

PRIMEROS PASOS

Debemos empezar a entrenar nuestra mente y forma de pensar a descubrir como funcionan las cosas y como se resuelven los problemas.

- Los algoritmos: La base de todo.

SESSION 2

DATOS.

- **Simples**
  - **numericos**
    - **enteros (1,2,34,5,5,6)**
    - **reales (1.2, 2.4, 2.1,2323.12)**
  - **logicos**
    - **true**
    - **false**
  - **caracter**
    - **un caracter**

- **Estructurados**
  - **cadenas (conjunto de caracteres),**
  - **vectores y matrices (a=[1,2,3])**
  - **registros**
  - **punteros**

IDENTIFICADORES Y PALABRAS RESERVADAS

- **IDENTIFICADOR:** nombre que se le da al programa, variables, constantes definidos por el programador.

  - **CONSTANTE:** contiene un valor que no cambia durante la ejecucion del programa.
  - **VARIABLE:** contiene un valor que cambia de acuerdo a la logica del programa definida por el programador.

  - **ASIGNACION:** Para que una variable o constante guarde un valor se debe usar el simbolo "=" para asignarle ese valor a la variable o constante.
    (variable = 0, constante= 10, variable = variable + constante)

- **PALABRA RESERVADA:** identificador definido por el lenguaje como palabra clave y no puede ser usado por el programador como identificador.

- **FUNCIONES INTERNAS:** Realizan operaciones comunes establecidas por el lenguaje

CABECERA DEL PROGRAMA

- **Nombre del programa**
- **Declaraciones de constantes, variables y tipos de datos**
- **Declaraciones de subprogramas**

algoritmo MiAlgoritmo
{Esto es un comentario}
constantes
constante1 = 10
variables
variable1 = 0
inicio
<cuerpo del programa>
fin

OPERACIONES ARITMETICAS

- **^ exponenciacion**
- *** multiplicacion**
- **/ division**
- **+ suma**
- **- resta**
- **div division entera (cociente)**
- **mod modulo (resto de la division)**

PRIORIDAD DE LAS OPERACIONES ARITMETICAS

- ^ mayor
- *** , /**
- **+,-**
- **div, mod** menor

Reglas de prioridad

- **Los operadores de igual prioridad se evaluan de izquierda a derecha**
- **Si una expresion contiene parentesis con subexpresiones estas se evaluan primero**

Operadores relacionales

- == igual
- <> diferente
- <= menor o igual que
- >= mayor o igual que
- > mayor que
- < menor que

ENTRADA/SALIDA

- Entrada desde un dispositivo de entrada
- Salida en la pantalla e impresora

Ejercicio.
Determinar los valores de a, b, c y d despues de la ejecucion del codigo

```
INICIO:
    a =1, b=4
    c = a+b
    d = a-b
    a = c + 2 * b
    b = c + b
    c = a * b
    d = b + d
    a = d / c
FIN
```

Encontrar el valor de

a) VALOR = 4.0* 5

b) x=3.0, y=2.0

VALOR = x^y-y

c) VALOR=5, x=3

VALOR=VALOR*x

1. Calcular el perimetro y superficie de un rectangulo dada la base y la la altura del mismo (S=BASE*ALTURA, P=2*(BASE+ALTURA))
2. Calcular la superficie de un circulo (s = pi * r^2)
3. Se tienen 3 variables A,B,C. Intercambiar los valores entre si de la forma: b toma el valor de a, c toma el valor b, a toma el valor de c. NOTA: Solo se puede usar una variable auxiliar.

SESSION 3

ITERACIONES

- **CONDICIONALES**
  - si entonces (alternativa simple)

si condicion
entonces accion1
fin_si

si condicion **entonces**
accion1
fin_si

  - si entonces sino (alternativa doble)

si condicion
entonces accion1
sino accion2
fin_si

si condicion **entonces**
accion1
sino
accion2
fin_si

  - segun o en caso (alternativa multiple)

segun expresion **hacer**
opcion:accion1
opcion:accion2
fin_segun

segun expresion **hacer**
opcion:accion1
opcion:accion2
en_otro_caso: accionX
fin_segun

CICLOS O BUCLES

- **DESDE (for)**

  desde(i=0, **hasta** 10, i=i+1**)**
  acciones
  fin_desde

- **MIENTRAS (while o do while)**

  mientras condicion
  acciones
  fin_mientras

  hacer
  acciones
  mientras condicion

- **REPETIR (repeat o do until)**

  repetir
  acciones
  hasta_que condicion

Ejercicios

Se desea obtener la nomina semanal (salario neto), de los programadores de una empresa cuyo trabajo se paga por horas y del siguiente modo:

- Las horas inferiores o iguales a 35 (normales) se pagan a una tarifa determinada que se debe introducir por teclado igual que el numero de horas y el nombre del programador
- Las horas superiores a 35 se pagaran como extras a un precio de 1.5 las horas normales
- Los impuestos a deducir a los programadores varian en funcion de su sueldo mensual.
  - Sueldo <= 20,000 libres de impuestos
  - Los siguientes 35,000 al 20%
  - El resto al 30%

Obtener el algoritmo que permita calcular cualquier tabla de multiplicar.
Desarrollar un algoritmo que determine en un conjunto de 100 numeros naturales introducidos por el usuario, ¿Cuantos son menores de 15, mayores de 50 y cuantos estan comprendidos entre 45 y 55?
Se necesita conocer una serie de datos de unas empresas con 50 programadores:

a) ¿Cuantas personas ganan mas de $300 al mes? (salarios altos)
b) ¿Entre 100 y 300? (salarios medios)
c) ¿Menos de $100? (salarios bajos y empleados a tiempo parcial)

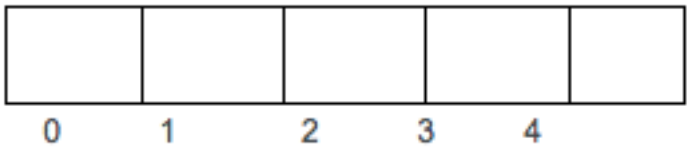
SESION 4

ESTRUCTURAS DE DATOS

- **VECTORES (ARRAYS)** Se identifica con su nombre y se le asocia con un nombre de variable, almacena datos del mismo tipo.

    - **<u>UNIDIMENSIONALES (LISTAS)</u>** Secuencia de elementos del mismo tipo y en los que el orden es significativo, el orden viene dado por un indice del array.

      entero nombre[n]
      donde n = numero de elementos



H = [16.5, 14.2, 5, 3.45, 0] donde I = 2

- **OPERACIONES**

Instruccion (accion)	Efecto
escribir (H[4])	Muestra 0 (valor de H[4])
escribir (H[I])	Muestra 5 (valor de H[I], I=2)
escribir (H[I]+2)	Muestra 7 (H[2] + 2)
escribir (H[I+2])	Muestra 0 (H[2+2] = H[4])
escribir (H[I-1])	Muestra 14.2 (H[2-1] = H[1])
escribir (H[2+1])	Muestra 3.45 (H[2+1] = H[3])
H[I--] = H(I)	Asigna 5 a H[1]
H[I] = H(I+1)	Asigna 3.45 a H[2]
H[I] = H(0)	Asigna 16.5 a H[2]

- **LECTURA**

```
desde(i=1 hasta 10 hacer)
        escribir(H[i])
fin_desde

i=0
mientras i<=10 hacer
        escribir(H[i])
fin_mientras
```

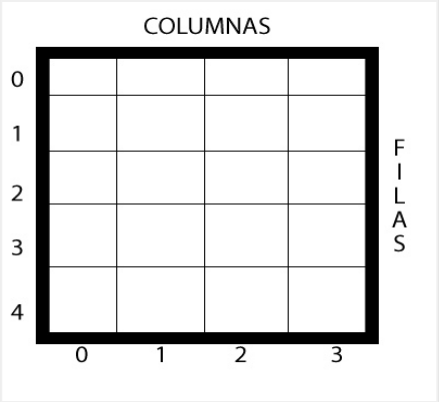
- **ESCRITURA**

```
desde(i=1 hasta 10 hacer)
        leer(H[i])
fin_desde

i=0
mientras i<=10 hacer
        leer(H[i])
fin_mientras
```

- ARRAYS BIDIMENSIONALES (vector de vectores) Conjunto de elementos del mismo tipo, con un orden significativo utilizando dos indices para definir cualquier elemento.

```
entero nombre[i][j]
    donde i = numero de elementos (filas)
            j = numero de elementos (columnas)
```



- LECTURA

```
desde(i=1 hasta 10 hacer)
        desde(j=1 hasta 10 hacer)
                escribir(H[i][j])
        fin_desde
fin_desde
```

- **ESCRITURA**

```
desde(i=1 hasta 10 hacer)
        desde(j=1 hasta 10 hacer)
                leer(H[i][j])
        fin_desde
fin_desde
```

Ejercicios:

Determinar la posicion del elemento mas grande de una tabla, donde la tabla es matriz[4][3]
Leer una matriz de tamaño 3 por 4 y escribirla en la pantalla

PYTHON

SESION 1

- Lenguage interpretado
- Tipado dinamico
- Fuertemente tipado
- Multiplataforma
- Orientado a Objetos
- ¿Por qué python?
- Herramientas basicas

RECURSOS:

- Instalacion http://www.youtube.com/watch?v=4Mf0h3HphEA

SESION 2

TIPOS BASICOS

- **int (todos los numero positivos y negativos), long (almacen numeros mas grandes)**

```
>>> numero = 3
>>> type(numero)
<type 'int'>

>>> numero = 3L
>>> type(numero)
<type 'long'>
```

- **float (contienen decimales)**

```
>>> real =10.3
>>> type(real)
<type 'float'>
```

- OPERADORES MATEMATICOS

Operador	Descripción	Ejemplo
+	Suma	r = 3 + 2 # r es 5
-	Resta	r = 4 - 7 # r es -3
-	Negación	r = -7 # r es -7
*	Multiplicación	r = 2 * 6 # r es 12
**	Exponente	r = 2 ** 6 # r es 64
/	División	r = 3.5 / 2 # r es 1.75
//	División entera	r = 3.5 // 2 # r es 1.0
%	Módulo	r = 7 % 2 # r es 1

- CONVERSION DE TIPOS DE DATOS

```
 >>> float(7.0)/2
3.5
>>> 7/2
3
>>> float(7)/2
3.5
>>> int(7.0)/2
3
>>> 7.0/2
3.5
>>> str(1)
'1'
```

- **CADENAS (texto encerrado entre 'cadena' o “cadena”)**

```
>>> c = 'cadena'
>>> cad = "cadena"
>>> c
'cadena'
>>> cad
'cadena'
>>> concatena = c+cad
>>> concatena
'cadenacadena'
>>> c*3
'cadenacadenacadena'
```

- Las cadenas se pueden manejar de acuerdo al encoding,existe unicode y ascci. Una cadena unicode se le antepone la letra u,ejemplo: u'mi cádena'

NOTA: la sentencia print solo imprime cadenas ascci,para imprimir una cadena no-ascci se debe convertir la cadena a ascci

```
>>> unicode = u'á éíóú'
>>> unicode
u'\xc3\xa1 \xc3\xa9\xc3\xad\xc3\xb3\xc3\xba'
>>> print unicode
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeEncodeError: 'ascii' codec can't encode characters in position 0-      1: ordinal not in range(128)
>>> print unicode.encode('latin-1')
á éíóú

>>> escapar_caracteres = "Esto es una \'cadena\' dentro de otra cadena"
>>> escapar_caracteres
"Esto es una 'cadena' dentro de otra cadena"
>>> print escapar_caracteres
Esto es una 'cadena' dentro de otra cadena

>>> cadena_multiple ="""
... Estoy escribiendo una cadena
... multiple, es con triple \'\'\' mi cadena \'\'\'
... """
>>> cadena_multiple
"\nEstoy escribiendo una cadena\nmultiple, ese con triple ''' mi cadena        '''\n"
```

- **BOOLEANOS**

OPERADOR	DESCRIPCION	EJEMPLO
and	¿se cumple a y b?	r = True and False # r es False
or	¿se cumple a o b?	r = True or False # r es True
not	No a	r = not True # r es False

```
>>>if a<b and c<b:
    print "true"

>>>if a<b or c>b:
    print "a y b es true"

>>>if not a>b:
        print "negacion"
```

- **OPERADORES RELACIONALES**
    - == igual
    - != diferente
    - <= menor o igual que
    - >= mayor o igual que
    - > mayor que
    - **< menor que**

RECURSOS:

- Numeros http://www.youtube.com/watch?v=YW8jtSOTRAU
- Variables http://www.youtube.com/watch?v=667ZeuZ0Q8M
- Cadenas http://www.youtube.com/watch?v=9v1HcKR39qw

SESION 3:

- **TIPOS DE COLECCIONES DE DATOS**

  LISTAS (equivalente al array en otros lenguajes y no necesariamente almacenan datos del mismo tipo)

```
>>> l = [12, False,"es cadena", [2,34]]
>>> l
[12, False, 'es cadena', [2, 34]]
>>> l[0]
12
>>> l[1]
False
>>> l[2]
'es cadena'
>>> l[3]
[2, 34]
>>> l[3][0]
2
>>> l[3][1]
34
>>> l[-4]
12
>>> l[-3]
False
>>> l[-2]
'es cadena'
>>> l[-1]
[2, 34]
>>> l[-1][-1]
34
>>> l[-1][-2]
2
```

  - **Particionado (selecciona partes de la lista)**
    - **(inicio:fin)** indica las posiciones del particionado sin contar el ultimo indice
    - **(inicio:fin:salto)** el tercer campo indica cada cuanto debe agregarse un elemento a la particion

```
>>> l
[12, False, 'es cadena', [2, 34]]
>>> l[1:3]
[False, 'es cadena']
>>> l[-1:-3] NOTA: Es al reves [-3:-1]
[]
>>> l[-3:-1]
[False, 'es cadena']
>>> l[:-1]
[12, False, 'es cadena']
>>> l[:]
[12, False, 'es cadena', [2, 34]]
```

```
>>> l[0:]
[12, False, 'es cadena', [2, 34]]
>>> l[1:]
[False, 'es cadena', [2, 34]]
>>> l[1:4:1]
[False, 'es cadena', [2, 34]]
>>> l[1:4:2]
[False, [2, 34]]
>>> l[1:4:3]
[False]
>>> l[-4:-1:3]
[12]
>>> l[-4:-1:2]
[12, 'es cadena']
>>> l[-4:-1:1]
[12, False, 'es cadena']
```

MODIFICANDO LA LISTA

```
>>> l[-4:-1:2] = [21, 'esto cambio']
>>> l
[21, False, 'esto cambio', [2, 34]]
```

- **TUPLAS (Es una lista inmutable, no se puede modificar ni cambiar de tamaño)**

```
>>> tupla = 1,True,[1,2]
>>> tupla
(1, True, [1, 2])
>>> tupla[2][0]
1
>>> tupla[2][1]
2
>>> tupla = 1,2,3,4,5
>>> tupla
(1, 2, 3, 4, 5)
>>> type(tupla)
<type 'tuple'>
>>> tupla[0]
1
>>> tupla[1]
2
>>> tupla[2]
3
>>> tupla[3]
4
>>> tupla[4]
5
>>> tupla[4] = 3
Traceback (most recent call last):
  File "<string>", line 1, in ?
TypeError: object doesn't support item assignment
```

DICCIONARIOS (matriz asociativa). Tienen un campo clave inmutable asociado a un valor. El acceso a los elementos del diccionario es usando el campo clave entre corchetes

```
>>> dicc = {'nombre': 'Juan', 'paterno':'perez', 'materno': 'Lopez'}
>>> dicc
{'paterno': 'perez', 'nombre': 'Juan', 'materno': 'Lopez'}
>>> dicc['nombre']
 'Juan'
  >>> dicc['paterno']
 'perez'
>>> dicc['materno']
'Lopez'
```

RECURSOS:

- Listas http://www.youtube.com/watch?v=XWQ0cyCrY7w
- Particionado http://www.youtube.com/watch?v=_IySULAqE_k

SESION 4:

- **ENTRADA / SALIDA**
  - **raw_input –** todo lo que ingreses lo regresa como cadena
    - Ingresando una cadena

```
>>> nombre = raw_input('Ingreso de datos: ')
Ingreso de datos: cadena
>>> nombre
'cadena'
>>> type(nombre)
<type 'str'>
```

    - Ingresando un entero

```
>>> nombre = raw_input('Ingreso de datos: ')
Ingreso de datos: 1
>>> nombre
'1'
>>> type(nombre)
<type 'str'>
```

    - Ingresando una lista

```
>>> nombre = raw_input('Ingreso de datos: ')
Ingreso de datos: [1,2,3,4,5]
>>> nombre
'[1,2,3,4,5]'
>>> type(nombre)
<type 'str'>
```

    - Ingresando flotantes

```
>>> nombre = raw_input('Ingreso de datos: ')
Ingreso de datos: 12.3
```

```
>>> nombre
'12.3'
>>> type(nombre)
<type 'str'>
```

- **Ingresando datos del tipo correcto**
  - Ingresando enteros

```
>>> nombre = int(raw_input('Ingreso de datos: '))
Ingreso de datos: 1
>>> nombre
1
>>> type(nombre)
<type 'int'>
```

  - Ingresando listas

```
>>> nombre = list(raw_input('Ingreso de datos: '))
Ingreso de datos: 12345
>>> nombre
['1', '2', '3', '4', '5']
>>> type(nombre)
<type 'list'>
```

  - Ingresando flotantes

```
>>> nombre = float(raw_input('Ingreso de datos: '))
Ingreso de datos: 12.3
>>> nombre
12.300000000000001
>>> type(nombre)
<type 'float'>
```

- **input –** regresa un objeto de acuerdo a su tipo.
  - Ingresando cadenas

INCORRECTO

```
>>> nombre = input('Ingreso de datos: ')
Ingreso de datos: Juan
Traceback (most recent call last):
 File "<stdin>", line 1, in <module>
 File "<string>", line 1, in <module>
NameError: name 'Juan' is not defined
```

CORRECTO

```
>>> nombre = input('Ingreso de datos: ')
Ingreso de datos: 'Juan'
>>> nombre
'Juan'
>>> type(nombre)
<type 'str'>
```

- Ingresando enteros

```
>>> nombre = input('Ingreso de datos: ')
Ingreso de datos: 1
>>> nombre
1
>>> type(nombre)
<type 'int'>
```

- Ingresando tuplas

```
>>> nombre = input('Ingreso de datos: ')
Ingreso de datos: 1,2,3,4,5
>>> nombre
(1, 2, 3, 4, 5)
>>> type(nombre)
<type 'tuple'>
```

- Ingresando listas

```
>>> nombre = input('Ingreso de datos: ')
Ingreso de datos: [1,2,3,4,5]
>>> nombre
[1, 2, 3, 4, 5]
>>> type(nombre)
<type 'list'>
```

- Ingresando diccionarios

```
>>> nombre = input('Ingreso de datos: ')
Ingreso de datos: {1:'a',2:'b'}
>>> nombre
{1: 'a', 2: 'b'}
>>> type(nombre)
<type 'dict'>
```

- **SALIDA**

print – Seguida de una cadena o en caso contrario de una variable. Estas cadenas permiten usar tecnicas de formateo.

Especificador	Formato
%s	cadena
%d	entero
%o	Octal
%x	Hexadecimanl
%f	Real

```
>>> print nombre
{1: 'a', 2: 'b'}
>>> real = 10.33345
>>> print "Numero real %f" %real
Numero real 10.333450
>>> entero =10
>>> print "Numero real %.2f y entero %i" %(real,entero)
Numero real 10.33 y entero 10
>>> print "Mi llamo %- 10s Lopez" % nombre
Mi llamo Juan       Lopez
>>> print "Mi llamo %10s Lopez" % nombre
Mi llamo       Juan Lopez
```

Ejercicios:

- Escribe un programa que pida el peso y la altura de una persona y que calcule su índice de masa corporal (imc). El imc se calcula con la fórmula imc = peso / altura^2.
- Escribe un programa que pida una distancia en pies y pulgadas y que escriba esa distancia en centímetros. Recuerda que un pie son doce pulgadas y una pulgada son 2,54 cm.
- Escribe un programa que pida una cantidad de segundos y que escriba cuántos minutos son.
- Escribe un programa que pida una cantidad de segundos y que escriba cuántas horas y minutos son.
- Escribe un programa que permita crear una lista de palabras de 5 elementos. Para ello, el programa tiene que solicitar ese número de palabras para crear la lista. Por último, el programa tiene que escribir la lista.

SESION 5:

Apartir de aqui todas las sentencias condicionales, ciclo, funciones, clases terminaran con dos puntos (:) y el codigo que le sigue estara indentado (mas a la derecha) al mismo nivel para indicar que le corresponde a la sentencia condicional, ciclo, funcion o clase.

- **SENTENCIAS CONDICIONALES – todas las sentencias teminan con dos puntos (:)**

  - **if (alternativa simple) La condicion siempre debe ser verdadera para que se ejecute una accion**

```
>>> mayor = 10
>>> menor = 5
>>> if menor > mayor:
...     print "falso, no imprimira nada"
...
>>> if menor < mayor:
...     print "verdadero, imprimio"
...
verdadero
```

  - **if...else (alternativa doble) Si la primer condicion no se cumpli pasa a ejecutar la segunda accion**

```
>>> if menor > mayor:
...     print "falso"
... else:
...     print "mayor es %s" % mayor
...
mayor es 10

>>> if not menor > mayor:
...     print "falso, pero la condicion se esta negando se convierte en verdadera"
... else:
...     print "mayor es %i" % mayor
...
falso, pero la condicion se esta negando se convierte en verdadera
```

  - **if...elif...elif...else (alternativa multiple anidada, por supuesto) Contiene una alternativa doble y dentro de esa misma anidar otras. El elif es como un sino.**

```
>>> MAYOR = 20
>>> if menor > MAYOR:
...     print "Falso"
... elif menor > mayor:
...     print "Falso"
... else:
...     print "El menor es %i" % menor
...
El menor es 5
```

NOTA: No preguntar por el switch, en python no existe el switch.

si quieres un switch programalo

SESION 6:

- **CICLOS – ejecutan codigo un numero repetido de veces.**
  - **While – usa una condicion y mientras se cumpla se ejecuta el codigo**

```
>>> contador = 0
>>> while contador < 5:
...     print "valor contador es: %i" % contador
...     contador = contador +1
...
valor contador es: 0
valor contador es: 1
valor contador es: 2
valor contador es: 3
valor contador es: 4

>>> contador = 10
>>> while not contador < 5:
...     print "valor contador es: %i" % contador
...     contador = contador - 1
...
valor contador es: 10
valor contador es: 9
valor contador es: 8
valor contador es: 7
valor contador es: 6
valor contador es: 5
```

  - **for...in – Se usa para recorrer secuencias. Lo que hace la cabecera del bucle es obtener el siguiente elemento de la secuencia** lista **y almacenarlo en una variable de nombre** elemento. **El for se lee "para cada elemento en secuencia"**

```
>>> lista = ['a',False, 2]
>>> for elemento in lista:
...     print elemento
...
a
False
2
```

  - **Un for como en c o php**

```
>>> for elemento in range(1,10):
...     print elemento
...
1
2
3
4
5
6
7
8
9
```

Ejercicios:

- **Escribe un programa que pida el año actual y un año cualquiera y que escriba cuántos años han pasado desde ese año o cuántos años faltan para llegar a ese año.**
- **Escribe un programa que pida dos números enteros y que calcule su división, escribiendo si la división es exacta o no.**
- **Escribe un programa que pida dos números y que escriba cuál es el menor y cuál el mayor o que escriba que son iguales.**
- **Escribe un programa que pida dos números enteros y que escriba si el mayor es múltiplo del menor.**
- **Escribe un programa que pida tres números y que escriba si son los tres iguales, si hay dos iguales o si son los tres distintos.**
- **Escribe un programa que pida la altura de un triángulo y lo dibuje de la siguiente manera:**
  Altura del triángulo: 4

  *****
  ******
  ****
  ****

SESSION :7

- **Funciones – fragmento de codigo asociado a un nombre y que regresa un valor. Permiten dividir el programa y no repetir codigo.**

    - Declaracion – usan la palabra def seguido del nombre de la funcion y terminan con 2 puntos (:)

```
>>> def mi_funcion():
...     print "mi primer funcion"
..
```

    - Ejecucion- mandando a llamara a la funcion

```
>>> mi_funcion()
mi primer funcion
```

    - Paso de parametros

```
>>> param1 = "primer parametro"
>>> param2 = "segundo parametro"
>>> def mi_funcion(param1, param2):
...     print param1, param2
...
>>> mi_funcion(param1,param2)
primer parametro segundo parametro
```

    - Inicializando parametros – el orden de los parametros no altero el resultado

```
>>> mi_funcion(param2 = "segundo parametro", param1 = "primer parametro")
primer parametro segundo parametro
```

    - El numero de parametros que se pasan al ejecutar la funcion deben corresponder a la misma cantidad de parametros en la defincion de la funcion

```
>>> mi_funcion()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: mi_funcion() takes exactly 2 arguments (0 given)
```

    - Salvo que en la definicion de la funcion los parametro este inicializados con None

```
>>> def mi_funcion(param1 = None, param2 = None):
...     print param1, param2
...
>>> mi_funcion()
None None
>>> mi_funcion(param1 = "primer parametro")
primer parametro None
```

- **Numero variable de argumentos por valor – se usa el simbolo * seguido del nombre del parametro, retornara una tupla de esta manera se obtiene el valor del elemento.**

```
>>> def mi_funcion(param1, param2, *nparams):
...     print param1, param2
...     for value in nparams:
...         print value
...
>>> mi_funcion("primer parametro", "segundo parametro", "tercer parametro", "cuarto parametro")
primer parametro segundo parametro
tercer parametro
cuarto parametro
```

- **Numero variable de argumentos por nombre – se usa el simbolo ** seguido del nombre del parametro, retornara un diccionario de esta manera se obtiene el valor del nombre para obtener el elemento.**

  - **Solo imprime la clave**

```
>>> def mi_funcion(param1 = None, param2 = None, **nparams):
...     print param1, param2
...     for value in nparams:
...         print value
...
>>> mi_funcion(param1, param2, param3="tercer parametro", param4="cuarto parametro")
primer parametro segundo parametro
param4
param3
```

  - **Imprimiendo el valor del diccionario que se genero al pasar n parametros.**

```
>>> def mi_funcion(param1, param2, **nparams):
...     print param1, param2
...     for value in nparams.items():
...         print value
...
>>> mi_funcion(param1, param2, param3="tercer parametro", param4="cuarto parametro")
primer parametro segundo parametro
('param4', 'cuarto parametro')
('param3', 'tercer parametro')

>>> def mi_funcion(param1, param2, **nparams):
...     print param1, param2
...     for value in nparams.items():
...         print value[1]
...
>>> mi_funcion(param1, param2, param3="tercer parametro", param4="cuarto parametro")
primer parametro segundo parametro
cuarto parametro
tercer parametro
```

BIBLIOGRAFIA

- **Problemas de metodologia de la programación**
  Luis Joyanes Aguilar
  McGraw Hill

- **http://www.mclibre.org/consultar/python/**
- **http://docs.python.org/**
- **http://www.mbmproject.com/tbtricks/python_modules.php**