El nucleo de la programación

Temario de la clase

- Cadenas de caracteres.
- Listas y tuplas.
- Variables lógicas. True False.
- Bifurcaciones. If else.
- Variables bandera (flag).

Cadenas de caracteres

Subcadenas [i:j]:

```
>>> sa='cadena'
>>> print sa[2:4]
de
```

Transforma un número en cadena, str

```
>>> a=1239
>>> b=str(a)
>>> print b[2:4]
39
```

Concatena una cadena: +

```
>>> nombre='Juan'
>>> apellido='Perez'
>>> nombre_completo=nombre+' '+apellido'
>>> print nombre_completo
Juan Perez
```

Operaciones con cadenas de caracteres

Cambia a mayúsculas: .upper()

```
>>> a=’casa’
>>> print a.upper()
CASA
```

Reemplaza un caracter o cadena de caracteres: .replace(viejo,nuevo)

```
>>> print a.replace(’a’,’o’)
coso
```

Busca un caracter o cadena de caracteres: .find(’o’)

```
>>> a=’casona’
>>> print a.find(’o’)
3
```

Multiplicación?

Tipos de variables: listas

Ya hemos visto los tipos: Enteros. Flotantes. Cadena de caracteres. Lógicas.

Pero también existen tipos de variables que son conjuntos de enteros, flotantes, etc...

Una lista es un conjunto de elementos de cualquier tipo separados por comas y delimitado por corchetes:

```
>>> lista=[25,60.4,'edad y peso']
```

Para acceder a un elemento de la lista:

```
>>> print lista[1]
```

Para acceder a varios elementos de la lista:

```
>>> print lista[1:2]
```

Cantidad de elementos de la lista:

```
>>> print 'Longitud: ',len(lista)
```

Si quiero cambiar la edad en la lista:

```
>>> lista[0]=26
```

Operaciones con listas

range(<inicio,>fin<,salto>): crea una lista de enteros, de uno en uno (el fin esta excluido!).

```
>>> print range(4)
[0, 1, 2, 3]
```

```
>>> print range(2,10,2)
[2, 4, 6, 8]
```

lista.append(elemento) Agrega elementos a una lista

```
>>> z = [1,2.02]
>>> z.append(800.8)
>>> z
[1, 2.02, 800.8]
```

Operaciones con listas

lista.reverse(): invierte la lista

```
>>> z.reverse()
>>> print z
[800.8, 2.02, 1]
```

lista.remove(x): elimina el primer elemento que coincide con x de la lista

lista.pop(j): elimina el elemento j-esimo de la lista

Tipos de variables: tuplas

Las tuplas son secuencias de objetos como las listas pero no se pueden cambiar

```
>>> tupla=[25,60.4,'edad y peso']
```

si engordo y quiero cambiar el peso:

```
>>> tupla[1]=61.6
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

La razón de su existencia es que son mucho mas eficientes sin embargo nosotros la usaremos muy poco (en general requerimos de estructuras mas dinámicas).

Variables lógicas

Una variable lógica puede tomar dos valores: True o False.

```
>>> lpreg=True
```

```
>>> type(lpreg)
<type 'bool'>
```

'bool'=boolean

Operaciones con variables lógicas

Existen tres operaciones de variables lógicas: and, or y not.

```
>>> lresp=False
```

Operador and

```
>>> lresp and lpreg
False
```

True and True $\rightarrow$ True

True and False $\rightarrow$ False

False and False $\rightarrow$ False

Operador or

```
>>> lresp or lpreg
True
```

True or True $\rightarrow$ True

True or False $\rightarrow$ True

False or False $\rightarrow$ False

Operaciones con variables lógicas

Operador not

```
>>> not lpreg
False
```

Not True $\rightarrow$ False

Not False $\rightarrow$ True

Operaciones combinadas (OJO con el orden!)

```
>>> lcom=lresp and (lpreg or not lbe)
>>> lcom False
```

Operadores que resultan en variables lógicas

Es la variable a igual a la variable b? ==

```
>>> lresp=a == b
```

Es la variable a distinta a la variable b? !=

```
>>> lresp= 1!=2
```

Es la variable a mayor a 5? >

```
>>> lresp= a > 5
```

```
>>> lresp= a >= 6
```

Combinación de operaciones:

```
>>> lresp= a >=6 and a <=10
```

El resultado de todas estas operaciones es una variable lógica. True False

Operador para cadena de caracteres: strings

```
>>> s1 = 'bc'
>>> s2 = 'abcde'
```

El operador in pregunta si una cadena se encuentra en la otra:

```
>>> s1 in s2
```

El operador in not pregunta si una cadena no se encuentra en la otra:

```
>>> s1 in not s2
```

El operador is pregunta si una variable es la otra (en muchos contextos es similar a ==, pero mas pythonic porque es legigle).

```
>>> x0 is 5
>>> x0 is None
```

Las variables las puedo definir como 'None'

La instrucción if: condicional

Hay muchas veces en un programa que vamos a querer controlar el flujo, es decir que el programa haga algo si la respuesta es afirmativa y que no lo haga si la respuesta es negativa:

```
>>> syes=raw_input("Desea terminar (s): ")
>>> if syes == 's':
...     print 'Respuesta s=si. Termino el programa'
...     quit()
```

La estructura de la instrucción if es:

if (variable lógica): Si la variable lógica es verdadera entonces:

(4 espacios en blanco) Haga esto

Los espacios en blanco, tabulación, son parte de la instrucción.

La instrucción if-else

Si pasa esto, haga algo si no pasa eso haga otra cosa:

```
syes=input("Desea continuar (s/n): ")
if syes == 's':
    print 'Respuesta s=si continua.'
else:
    print 'Cualquier otra respuesta termina. '
    quit(). '
```

La estructura de la instrucción if-else es:

if (variable lógica): Si la variable lógica es verdadera entonces:

(4 espacios en blanco) Haga esto

else: Si la variable lógica es falsa entonces

(4 espacios en blanco) Haga esto otro

La instrucción if-else. Ejemplos.

Si queremos calcular raíces cuadradas a partir de un número que introduce el usuario, nos deberíamos asegurar que los números son positivos para que no haya error.

```
a=input('Introduzca el nro: ')
if a > 0:
    sqa=math.sqrt(a)
    print 'La raiz cuadrada del nro es:',sqa
else:
    print 'El nro debe ser positivo'
```

La instrucción if-else. Ejemplos.

El resultado lo podemos guardar en una variable lógica y luego usar la variable en el if.

```
a=input('Introduzca el nro: ')
lapos=a > 0
if lapos:
    sqa=math.sqrt(a)
    print 'La raiz cuadrada del nro es:',sqa
else:
    print 'El nro debe ser positivo'
```

Varias opciones elif.

Hay veces que necesitamos varias opciones no solo dos. Para esto existe el elif. Es una mezcla de else y de if, de lo contrario si pasa esto....

```
a=input('Introduzca un nro: ')
if a == 0:
    print 'El nro es zero'
elif a> 0:
    print 'El nro es positivo'
else:
    print 'El nro es negativo'
```

Varias opciones elif. Ejemplo.

El elif es útil para cuando se le da opciones al usuario [No existe el case].

```
a=input('Introduzca un nro: ')
print 'Que desea calcular: '
opt=input('(1) Cuadrado, (2) Raiz cuadrada, (3) Logaritmo
if opt == 1:
    print 'El cuadrado es: ',a**2
elif opt == 2:
    print 'La raiz es: ',math.sqrt(a)
elif opt == 3:
    print 'El logaritmo es: ',math.log(a)
else:
    print 'Hay solo tres opciones 1,2,3'
```

En estos casos siempre conviene usar un else a lo último para cualquier problema que hubo en el ingreso de los datos (o cuando se esta ejecutando el programa),

Entonces estamos avisando de que “No se encontró ningun opcion válida”.

Variables bandera.

En numerosas situaciones queremos guardar el estado de una situación.
Generalmente la variable bandera o flag tiene dos opciones 0 o 1.
En algun momento guardamos el estado de situación:

```
if nro % 2 == 0:
   band=1
else:
   band=0
```

En otro lugar del programa usamos el estado de situación de la variable:

```
if band == 1:
   print 'El numero es par'
```