

Conceptos generales de Python

Instrumentación básica y robótica [2024-2]

Miguel Angel Robles R.

ENCiT

Instrumentación básica y robótica

- Es un lenguaje interpretado
- Puede ejecutarse de dos maneras
 - ▶ Modo Interactivo
 - ▶ Modo script

- Se lanza mediante el comando `python` (o `python3`) desde el emulador de terminal
- Cada línea se ejecuta cuando el usuario la escribe seguida de *enter*
- Es útil para probar fragmentos de código o verificar instrucciones
- El prompt se compone de 3 símbolos "mayor que": `>>>`
- Para conocer el valor de una variable, sólo debemos escribir su nombre seguido de *enter*
- *help* puede ser de ayuda

Tipos de datos básicos

Veamos algunos ejemplos

Tipo	Ejemplo
Enteros	1, 2, 3,
Flotantes	3.14, 0.33, 5e-3
Complejos	2 + 3j
Booleanes	True, False, 1, 0

Operadores Básicos

Operador

Es un símbolo que representa una operación

Ejemplos de operadores

Tipo	Operadores
Aritméticos	+, -, *, /, %, //, **, (), =, +=, -=, ...
Lógicos	and, or, not
Comparación	==, >, <, <=, >=

Realiza las operaciones indicadas en modo interactivo

- Multiplicar 14 por 8
- Obtener el cuadrado de 9
- 2 elevado a 10
- el módulo entre 10 y 3
- Obtener la parte entera de la división entre 23 y 7
- Evaluar si 13×14 es menor que $160 + 22$

Nombres de variables

- Debe empezar con una letra o con '_'
- Puede continuar con letras, números o '_'
- Puede contener ñ o caracteres acentuados
- Case sensitive
- No debe
 - ▶ Contener espacios
 - ▶ Ser palabras reservadas
- Recomendable
 - ▶ Describa el contenido
 - ▶ No ser excesivamente largo

Palabras Reservadas

- False
- None
- True
- and
- as
- assert
- break
- class
- continue
- def
- del
- elif
- else
- except
- finally
- for
- from
- global
- if
- in
- is
- import
- lambda
- nonlocal
- not
- or
- pass
- raise
- return
- try
- while
- with
- yield

Nombres de Variables (Ejercicio)

Para la siguiente lista de variables indique cuales tienen nombres incorrectos e indique una corrección

- ❶ Parangaricutirimicuarro
- ❷ cloronitrotetramincobalto
- ❸ año
- ❹ árbol_1
- ❺ atmósfera
- ❻ Δt
- ❼ color_azul
- ❽ Print
- ❾ Nuevo León

Variables

- En python las variables no se declaran
- Las variables son del tipo del valor que se les asigne
- Su tipo puede cambiar

Ejemplos:

```
a= 1
```

```
a= 3.14
```

```
a= "texto"
```

```
a= True
```

Tipos de datos Compuestos

Son variables que agrupan datos más simples, en Python tenemos:

- Listas
- Tuplas
- Diccionarios

- Mantienen ordenados (por medio de un índice) diferentes valores
- El primer índice es 0
- Son mutables
- Se declaran como una lista de elementos encerrada por corchetes y separados por comas
- Se agregan elementos con el método "append"

Ejemplos:

```
pares = [2,4,6,8 ]  
letras = ['a', 'b','c']  
fecha = [1, 'octubre', 2014]  
pares.append(10)
```

Listas (acceso a elementos)

Podemos obtener el valor (y tipo) a sus miembros por medio de su índice

```
pares[0]  
pares[-1]
```

Podemos re-asignar el valor (y tipo) a sus miembros por medio de su índice

```
pares[0] = 1  
letras[1] = 1  
fecha[-1] = 2013
```

Una lista puede contener otras listas

```
fecha[-1] = [2014]
```

Concatenación de listas

Para concatenar listas se usa el símbolo "+". Ejemplos:

```
pares = [2, 4, 6, 8]
impares = [1, 3, 5, 7, 9]
digitos = pares + impares
```

Dividiendo listas

Podemos acceder o re-asignar una parte de la lista indicando el intervalo deseado, para esto se colocan los índices inicial y final de la sublista separadas por ":".

Ejemplos:

```
digitos[0:2]
```

```
digitos[0:2]=[0,1,2,3]
```

```
digitos[0:2]=[0]
```

Operaciones con listas

Operación	Descripción
<code>s[i]=x</code>	Elemento i es reemplazado con x
<code>s[i:j]=t</code>	Parte de i a j es reemplazado con lista t
<code>del s[i:j]</code>	Elimina los elementos de i a j
<code>s.append(x)</code>	agrega x al final de s
<code>s.extend(t)</code>	extiende s con los elementos de t
<code>s.insert(i, x)</code>	inserta x in s en el lugar i
<code>s.pop([i])</code>	obtiene el elemento i y lo remueve de s
<code>s.remove(x)</code>	elimina el primer elemento de s donde aparece x
<code>s.reverse()</code>	invierte la lista s
<code>len(s)</code>	devuelve el tamaño de s
<code>sort(s)</code>	ordena s
<code>x in s</code>	regresa <i>True</i> si existe x en s
<code>s.index(x[, i[, j]])</code>	regresa el índice de la primer ocurrencia de x

- Son secuencias inmutables (tanto en valor como en tamaño)
- Se define por medio de paréntesis
- Son más eficientes que las listas

Ejemplos:

```
pares = (2, 4, 6, 8)
```

```
impares = (1, 3, 5, 7, 9)
```

- Son una lista asociativa.
- Se conforman de pares de claves y valores
- Las claves deben ser inmutables
- No tienen orden
- Se escriben entre `{ y }`
- Cada par se separa por coma
- Cada pareja se asocia mediante dos puntos

Ejemplo:

```
edad = {'Alice':38, 'Bob':39, 'Dave':34}
```

Acceso y métodos

Se accesa mediante: [clave]

Ejemplo: edad[Bob]

in:

'Alice' in edad

algunos métodos:

- .values()
- .keys()
- .items()

Ejemplo:

edad.values()

Ejemplo

Crear un diccionario que contenga los siguientes textos: “menu”, “archivo”, “salir”. Las claves de los valores serán la primer letra de cada opción.

Respuesta

```
m = {'m':'menu', 'a':'archivo', 's': 'salir'}
```

Entrada de datos

Sintaxis

```
input([prompt])
```

Acción

Regresa la cadena que el usuario ingrese.

Parámetros

- prompt - Cadena a mostrar

Ejemplo:

```
input("Ingresa tu nombre")
```

Salida de datos

Sintaxis

```
print(*objects, sep=' ', end='\n', file=sys.stdout)
```

Acción

Produce la salida de datos.

Parámetros:

- *objects* - son las variables de salida.
- *sep* - cadena de separación
- *end* - cadena de fin de línea
- *file* - objeto de salida

Salida de datos (ejemplos)

```
print (m, edad)
print (m, edad, sep= '<->')
print (edad, sep= '\n')
print (m, edad, sep= '\n')
```

Modo Script

- Este modo permite ejecutar un programa desde un archivo
- Se ejecuta escribiendo python(o python3) seguido por un espacio y el nombre del archivo.
- En windows hay que ir a la carpeta donde se instaló python3
- En linux se puede ejecutar desde cualquier lugar
- Los comentarios van precedidos por # o encerrados entre tres comillas simples

Ejemplo:

```
'''Ejemplo de menu'''  
menu = {'m':'menu', 'a':'archivo', 's': 'salir'}  
menu_var = input('Elige opción: ')  
print(menu[menu_var])
```


Sentencia de selección (condición)

Selecciona el bloque de instrucciones a ejecutar dependiendo si se cumple alguna condición

Sintaxis

```
if condicion :  
    instruccion1  
    instruccion2  
    ...  
elif condición:  
    instruccion1  
    instruccion2  
    ...  
else:  
    instruccion1  
    instruccion2  
    ...
```

• *elif y else son opcionales*

Sentencia de Repetición

Repite un bloque de instrucciones mientras se cumpla la *condición*

Sintaxis

```
while condicion:  
    instruccion1  
    instruccion2  
    ...
```

- Los límites de la sentencia (bloque de instrucciones) se indican con el sangrado
- Se recomienda sangrado de 4 espacios

- Son una forma de representar texto
- Podemos verlas como una lista de caracteres
- Se puede representar caracteres especiales con la secuencia correspondiente:

Secuencia	Caracter
\\	\
\'	'
\"	"
\r	retorno de carro
\n	nueva línea
\t	tabulador

Cadenas (métodos)

Algunos métodos útiles son los siguientes

Método	Resultado
<code>x in s</code>	True si x está en s
<code>s*n</code>	realiza n copias de s y las concatena
<code>len(s)</code>	Longitud de la cadena s
<code>s.count(x)</code>	Incidencias de x en s
<code>s.find(x)</code>	El índice de la primer ocurrencia de x en s
<code>s.split(sep=None)</code>	Una lista con las palabras divididas por sep
<code>s.strip([chars])</code>	elimina los caracteres indicados en chars
<code>s.format()</code>	Realiza una operación de formateo

for

Permite recorrer un elemento iterable

Sintaxis

```
for <elemento> in <iterable>:  
    <bloque de código>
```

Ejemplo

```
lista = ['elemento', 10.5, 3 ]  
for item in lista:  
    print(item)
```

```
elemento  
10.5  
3
```

Definición

Archivo es una sucesión de bytes en un dispositivo de almacenamiento (disco duro, CD, DVD, etc).

Tipos

- Binarios: Archivos cuya representación se encuentra en código binario
- Texto: Archivos cuya representación corresponde a una codificación de texto (por ejemplo utf-8)

Archivos (abrir y cerrar)

```
file = open(filename, mode)
```

Crea un objeto de tipo archivo y lo abre en el modo especificado

- file: Objeto de tipo archivo
- filename: Nombre de archivo
- mode: Modo, por defecto es 'r'
 - ▶ 'r' modo de lectura
 - ▶ 'w' modo de escritura, si el archivo existe lo sobrescribe
 - ▶ 'a' modo de concatenación

```
file.close()
```

Cierra el objeto de tipo archivo

- file: objeto de tipo archivo

Leyendo el contenido de un archivo

```
data = file.read(n)
```

Lee n datos de un objeto tipo archivo

file: objeto tipo archivo

data: cadena correspondiente al archivo

ejemplo

```
archivo=open('datos/testfile.cca')
```

```
datos=archivo.read()
```

```
archivo.close()
```

```
print(datos)
```

```
data = file.readline()
```

Lee una línea del archivo

Leyendo el contenido de un archivo (II)

Archivo a una lista

```
data = file.readlines()  
data = list(file)
```

Más eficientemente

```
for line in file:  
    print(line)
```

file.write(string)

Escribe una cadena en el archivo en el archivo

file Objeto de tipo archivo

string Cadena de texto

Leer un archivo de datos y extraerlos