POCKET DE PYTHON (PP)

Python es un Lenguaje de alto nivel , interpretado , creado a fines de los 80 por el profesor holandés Guido Van Rossum. Es abierto, multiplataforma y multiparadigma. Emplea tipado dinámico.

GENERALES DEL LENGUAJE

- Todo lo que precede en una línea al caracter # es un comentario, no es una instrucción ejecutable
- Todo lo que encierran los caracteres ''' ''' es un comentario de una o más líneas
- Para cortar condiciones o expresiones largas (>80 caracteres), se encierra entre ( ) y se corta en más de un renglón
- Las instrucciones que requieren ser escritas en más de un renglón se organizan o identifican a través del uso de la Indentación de Renglones
- Existen Palabras Reservadas de Python, es decir, no pueden usarse para denominar objetos propios
- El caracter de escape es \
  Ej: print('El programa imprime \'Hola Mundo\'')
  Muestra: El programa imprime 'Hola Mundo'

GENERALES DE LA PROGRAMACIÓN PROFESIONAL

- Los programas se escriben en letras minúsculas
- Los nombres de variables se escriben en minúsculas y se usan nombres nemotécnicos
  Ej:
  cantidad
- Si es necesario usar una frase, se la escribe sin separaciones y en mayúscula las iniciales de cada palabra, excepto la primera
  Ej:
  cantidadDeUso
- Los nombres de constantes se escriben en mayúscula
  Ej:
  VERDADERO

GENERALES DE ESTE MANUAL:

Todo lo que se muestra en **negrita** en este manual es una palabra reservada de Python.
Ejemplo de Palabras Reservadas:
and-def-except-from-in-or-return-as-del-exec-global-is-pass-try-assert-elif-finally-if-lambda print-while-class-else-for-import-not-raise-with-yield

HERRAMIENTAS ALGORÍTMICAS

Operadores

Aritméticos: + , - , *, ** (potencia), /, // (cociente entre enteros), % (resto)
Comparación: < , > , <= , >= , == (igual), != (distinto)
Lógicos: and , or , not
Conjunto: in,|,&,-,^
Secuencia: in
Prioridades o Precedencias:: 1: ( ) 2: not 3: *, /, and 4: +, -, or 5: ==, !=, <, >, >=, >=, in,|,&,-,^

Asignaciones

Simple: *<variable> = <expresión>*
Compuesta: *<var1>,<var2>,..<varN> = <expr1>,<expr2>,…<exprN>*

expresión_simple:: constante|variable

expresión_compuesta:: combinación de *expresión_simple o expresión_compuesta* usando *operadores*

Entrada/Salida

Función de lectura o entrada para ingresar los datos por teclado (entrada estándar)
input::
input ([<*mensaje*>])

mensaje:: cualquier texto válido

Para almacenar en memoria el dato ingresado debe ser asignado

Ej:
descripcionArticulo = **input** ('Ingrese la Descripción del Artículo : ')

Para mostrar el valor de una variable en el mensaje de un input, se la puede convertir a string y unir al mensaje concatenando antes o entre los paréntesis

Otra forma es con formateo:

Ej:
input('mensaje %d resto del mensaje: ' %variableEntera) # %s para String

%f para real

Función de impresión o salida por pantalla (salida estándar)
print::
print(*<expresión_1>,<expresión_2>,…,<expresión_k>*, **end**=*<car_1>*, **sep**=*<car_2>)*

expresión_i:: cualquier string válida

car_i:: cualquier carácter válido

Condicional

if::
if <*condición_1*> :
<*cuerpo_1*>
[(**elif** <*condición_k*>:
<*cuerpo_k*>)*
else:
<*cuerpo_s*>]

cuerpo_i:: (cualquier sentencia válida de Python**)**$^{+}$

condición_i:: cualquier condición válida de Python

Ciclos

Genérico:

while::
while <*condición*> **:**
<*cuerpo*>

Sólo para repeticiones exactas :

for::
for <variable> **in** <*secuencia*> :
<*cuerpo*>

secuencia ::rango|conjunto|objetos de la Clase

Funciones de Casteo

int(x)	Convierte a entero x, x puede ser real o cadena	int(90.5)/ int('90')	90/90
float(x)	Convierte a real x, x puede ser entero o cadena	float(90)/float('90')	90.0/90.0
str(x)	Convierte a cadena x, x puede ser entero o real	str(90)/str(90.0)	'90'/'90.0'
bool(x)	Convierte a lógico x, x puede ser numérico o cadena	bool(0 oVacío)/ bool(1o char)	True/ False
chr(x)	Devuelve el carácter ascii del valor x	chr(65)	'A'
ord(x)	Devuelve el valor ascii del carácter x	ord('A')	65
list(x)	Devuelve una lista con los elementos de x (x debe ser objeto estructurado)	list('Hola')	['H','o','l', 'a']
tuple(x)	Devuelve una tupla con los elementos de x (x debe ser objeto estructurado)	tuple([1,2,3])	(1,2,3)
set(x)	Devuelve un conjunto con los elementos de x, x puede ser cualquier secuencia	set([1,2,3])	{1,2,3}
type(x)	Devuelve el tipo de dato de x	type('Hola')	str

Funciones y Métodos Predefinidos para Números

abs(x)	Devuelve el valor absoluto de x	abs(-5)	5
cmp(x,y)	Devuelve un valor <, =, > que cero resultante	cmp(4,4)	0
complex(x,y)	Devuelve el numero complejo x+yj	complex(3,5)	(3+5j)
a.real	Devuelve la parte real del complejo	a=complex(3,5) a.real	3.0
a.imag	Devuelve la parte imaginaria del complejo	a=complex(3,5) a.imag	5.0
divmod(x,y)	Devuelve la pareja cociente , resto entre x e y	divmod(7,3)	(2,1)
pow(x,y)	Devuelve la potencia y de x	pow(3,2)	9
round(x,n) round(x)	Redondea x a n dígitos tras el punto decimal Redondea a x al entero más próximo	round(34.59,1) round(34.59999)	34.6 35

Bibliotecas random y math (import random , import math)

random.choice(valores)	Devuelve un aleatorio perteneciente a valores
random.randint(a,b) random.uniform(a,b)	Devuelve un aleatorio entero entre a y b Devuelve un aleatorio real entre a y b
random.random()	Devuelve un aleatorio real entre 0 y 1
math.pi math.e	Devuelve el valor de la constante pi Devuelve el valor de la constante e
math.exp(x)	Devuelve e^x
math.sqrt(x)	Devuelve la raíz cuadrada de x
math.log(x) log10(x)	Devuelve el logaritmo natural de x Devuelve $\log_{10}$ de x
math.sin(x) math.cos(x) math.tan(x)	Devuelve el seno de x Devuelve el coseno de x Devuelve la tangente de x
math.ceil(x) math.floor(x)	Devuelve techo de x Devuelve piso de x

Funciones Propias

Definición

Definir todas las funciones propias al principio del programa, una a continuación de otra y en cualquier orden

Definición Función Propia::
def <*nombre*> ([*param_1,…,param_k*]):
    <*cuerpo*>

Puede devolver o no valores usando **return** <*salida*> como última sentencia ejecutable del cuerpo

salida:: dato simple o estructurado

Invocación

Pueden ser invocadas desde cualquier parte del programa o desde otro subprograma definido en él

El ámbito de invocación dependerá de si devuelve o no valores

Se invoca por el nombre y se le envían los parámetros esperados en el orden esperado. El orden se puede alterar si se especifican pares *nombre_de_parámetro=valor*

ESTRUCTURAS DE DATOS

Tipos de Datos Simples

Enteros - Constante entera: 35 609 -12
Reales o Flotantes - Constantes reales: -6.8 15.0
Imaginarios - Constante imaginaria: (10,9j)
Lógicas o Booleanas - Constante booleana: True False
Caracteres - Constante carácter: '1' 'F'

Clase Conjunto

Funciones y Operaciones para la Clase Conjunto

x **in** c	Devuelve **True** si x pertenece a c, **False**, en caso contrario.
x not in c	Devuelve **True** si x no pertenece a c, **False**, en caso contrario.
c|b^+	Devuelve la unión de c y b. Puede haber más de dos conjuntos.
c&b^+	Devuelve la intersección de c y b. Puede haber más de dos conjuntos.
c-b	Devuelve la diferencia de c menos b.
c^b	Devuelve la diferencia simétrica entre c y b.
len(c)	Devuelve la cantidad de elementos de c.
min(c)	Devuelve el mínimo elemento de c.
max(c)	Devuelve el máximo elemento de c.

Clase Secuencia

Subclases de Secuencia: Rangos, String, Listas, Archivos

Funciones, Operaciones y Referencias de Elementos para la Clase Secuencia

x **in** s	Devuelve **True** si x pertenece a s, **False**, en caso contrario
s+t	Concatena la secuencia s y la t en ese orden
s*n	Concatena n veces la secuencia s
s[i] s[-k]	Referencia el elemento de la posición i de la secuencia s Referencia el elemento que está k posiciones antes del último
s[i:j]	Referencia la porción de la secuencia s que va del elemento i al j-1
s[i:j:k]	Referencia la porción de la secuencia s que va del elemento i al j-1, con paso k
len(s)	Devuelve la longitud de la secuencia s
min(s)	Devuelve el mínimo elemento de s
max(s)	Devuelve el máximo elemento de s

Rangos

range (*n*) secuencia de enteros entre 0 y *n-1* Ej: range(2) (0,1)
range (*n1, n2*) secuencia de enteros entre *n1* y *n2-1* Ej: **range**(2,5) (2,3,4)
range (*n1, n2, paso*) secuencia de enteros entre *n1* y *n2-1*, con frecuencia *paso*
Ejs:
range(6,13,2) (6,8,10,12)
range(14,6,-2) (14,12,10,8)

Clases Inmutables

Cadenas o String

s='' Cadena vacía, longitud 0
s=' ' Cadena con un espacio, longitud 1
eval(*cadena*) intenta evaluar *cadena* como una expresión aritmética. Ej: **eval**('2+4') devuelve 6

Tuplas

(s1,…,sk) Pueden contener cualquier tipo de datos, simple o estructurado

Clases Mutables

Listas

lista[] Creación de una lista vacía, cero elementos
lista[i]= x Asigna x al elemento existente de la posición i de lista
sorted(lista[,reverse=**True**][,**key**=*función*]) devuelve lista ordenada. Si se indica **reverse** en orden descendente, si se indica **key** el criterio de ordenamiento lo indica *función*

Archivos

archivo = ***open****(nombre[,modo])* abre un archivo relacionando nombre físico con lógico

modo:

'r' sólo lectura (produce error si no existe). Este es el modo por omisión

'w' sólo escritura (lo sobreescribe si existe o lo crea si no existe)

'a' sólo escritura al final (se posiciona al final o lo crea si no existe, y permite agregar)

'r+'/'w+'/'a+' lectura y escritura, creándolo si no existe, lo sobrescribe si existe, lee desde el principio o escribe al final

linea = archivo.***readline()*** devuelve una línea del archivo y la carga en la string *linea*

linea = ***linea.rstrip("\n")*** remueve el fin de línea colocado por el Sistema Operativo

lineas = archivo.**readlines**() devuelve todas las líneas del archivo. Casi no se usa ***enumerate****(archivo)* enumera las líneas del archivo

*archivo.****write****(string+'\n')* # escribe el string en una línea del archivo, agrega carácter de final de línea de texto

*archivo.****writelines****(lista_de_cadenas)* escribe varias líneas del archivo

*archivo.****close****()* cierra el archivo

Ejemplo:

```
arch = open('datos.txt') # apertura y
                         asignación de solo lectura a la variable arch del archivo físico datos.txt
arch.readline() # esta primera lectura es para descartar el encabezado (si lo hubiera) de la primera línea
for linea in arch: # lectura y procesamiento de cada línea con datos del archivo
    linea = linea.rstrip("\n") # remueve el fin de línea colocado por el Sistema Operativo
    lista = linea.split(separador) # devuelve una lista con las palabras separadas
                por el separador de la línea
    # procesamiento de lista, registro o fila del archivo
arch.close() # cierre del archivo
```

Métodos de la Clase set (conjunto)

c.***add(****elemento****)*** agrega *elemento* a c. Sólo si *elemento* no pertenecía ya a c.

c.***pop()*** devuelve un elemento arbitrario de c y lo quita del conjunto.

c.***remove****(elemento)* elimina *elemento* de c. Si *elemento* no pertenece a c, da un error.

c.***discard****(elemento)* elimina *elemento* de c.

c.***clear****()* elimina todos los elementos de c y lo deja vacío.

c.***copy****()* devuelve una copia de c.

c.***isdisjoint****(b)* devuelve True si intersección entre c y b es vacía(son conjuntos disjuntos).

c.***issubset****(b)* devuelve True si c es subconjunto de b (todos los elementos de c pertenecen a b).

c.***issuperset****(b)* devuelve True si c contiene a b (todos los elementos de b pertenecen a c).

c.***union****(b_1[,b_k]*)* devuelve la unión entre c y b.Puede haber más de un argumento.

c.***update****(b)* c pasa a ser c|b.

c.***difference****(b)* devuelve la diferencia de c menos b.

c.***difference_update****(b)* c pasa a ser c-b.

c.***intersection****(b_1[,b_k]*)* devuelve la intersección entre c y b.Puede haber más de un argumento.

c.***intersection_update****(b)* c pasa a ser c&b.

c.***symmetric_difference****(b)* devuelve la diferencia simétrica entre c y b (los elementos de c y b que no pertenecen a la intersección de ambos).

Métodos de la Clase String

s.***capitalize()*** devuelve una copia del string con la primera letra en mayúscula y el resto en minúscula
s.***center(****ancho[,relleno]****)*** string centrado con ese relleno a los costados
s.***count****(substring[,desde[,hasta]])* devuelve la cantidad de veces que aparece el substring en el string
s.***find****(substring[,desde[,hasta]])* devuelve la primera posición de comienzo del substring en s
s.***rfind****(substring[,desde[,hasta]])* devuelve la última posición de comienzo del substring en s
s.***format****(fmtstr,*args,**kwargs)* devuelve s formateada sustituyendo dinámicamente un texto

Ejemplo1:
```
texto="Bienvenido a mi aplicación{0}"
print(texto.format(" en Python"))
```
Retorna:
```
Bienvenido a mi aplicación en Python
```

Ejemplo2:
```
texto="Bruto: ${bruto} + IVA: ${iva} = Neto: ${neto}"
print(texto.format(bruto=100,iva=21,neto=121))
```
Retorna:
```
Bruto: $100 + IVA: $21 = Neto: $121
```

s.***index****(substring[,desde[,hasta]])* idem a s.find
s.***rindex****(substring[,desde[,hasta]])* idem a s.rfind
s.***join****(iterable)* arma una string uniendo los elementos de iterable e intercalándolos con s

Ejemplo:
```
tup=('a','b','c')
print('-'.join(tup))
```
Retorna:
```
a-b-c
```

s.***ljust****(ancho[,relleno])* justifica hacia la izquierda
s.*rjust(ancho[,relleno])* justifica a derecha
s.***lower()*** devuelve s en minúsculas
s.***upper()*** devuelve s en mayúsculas
s.***maketrans****(x[,y[,z]])* asocia en un diccionario los correspondientes ASCII de las cadenas x e y

Ejemplo:
```
vocales="aeiou"
numeros="12345"
texto="murcielagos"
print(texto.maketrans(vocales,numeros))
```
Retorna:
```
{97: 49, 111: 52, 117: 53, 101: 50, 105: 51}
```

s.***translate****(pares)* devuelve s con los caracteres asociados en el diccionario pares remplazados

Ejemplo:
```
vocales="aeiou"
acentos="áéíóú"
texto="murcielagos"
parejas=texto.maketrans(vocales,acentos)
print(texto.translate(parejas)
```
Retorna:
```
múrcíélágós
```

s.***partition****(separador)* Crea particiones de tupla a la izquierda
s.***rpartition****(separador)* Crea particiones de tupla a la derecha
s.***replace****(antes,ahora[,cantidad])* Reemplaza el substring de antes por el de ahora
s.***strip****()* elimina los espacios del inicio y fin del string
s.***lstrip()***elimina los espacios del inicio
s.***rstrip()*** elimina los espacios del fin
s.***swapcase()*** devuelve s con las mayúsculas convertidas en minúsculas y viceversa
s.***split****([separador[,maximaDivision]])* devuelve una lista cuyos elementos son las partes del texto separadas por separador. Si se omite separador toma blancos
s.***rsplit****([separador[,maximaDivision]])*ídem a derecha
s.***splitlines****([keepends])* Convierte el string en lista
s.***startwith****(prefijo[,desde[,hasta])* devuelve True si s comienza con prefijo, si no False

s.***endwith****(sufijo[,desde[,hasta])* devuelve **True** si s termina con sufijo, si no **False**
s.***zfill****(ancho)* Rellena con ceros a la izquierda hasta el ancho
s.***title()*** devuelve s en minúsculas con cada palabra inicializada en mayúsculas
s.***isnumeric()*** devuelve **True** si s es numérico, si no **False**
s.***isalnum()*** devuelve **True** si s es alfanumérico, si no **False**
s.***isalpha()*** devuelve **True** si s es alfabético, si no **False**
s.***isdecimal()*** devuelve **True** si s es decimal, si no **False**
s.***isdigit()*** devuelve **True** si s es un dígito, si no **False**
s.***isprintable()*** devuelve **True** si s es un carácter imprimible, si no **False**
s.***isspace()*** devuelve **True** si s es espacio, si no **False**
s.***istitle()*** devuelve **True** si s es un título, si no **False**
s.***isidentifier()*** devuelve **True** si s es un identificador (variable), si no **False**
s.***isupper()*** devuelve **True** si s está en mayúsculas, si no **False**
s.***islower()*** devuelve **True** si s está en minúsculas, si no **False**

Métodos de la Clase Listas

lista.***append****(valor)* agrega el elemento *valor* al final de la lista
lista.***insert****(posic,valor)* inserta el elemento *valor* en la posición *posic*
lista.***remove****(valor)* quita de la lista el elemento *valor*
lista.***pop****([índice])* quita de la lista el elemento de la posición *índice*. Si no se usa este parámetro, quita el último elemento
lista.***extend****(otraLista)* agrega al final de *lista otraLista*
lista.***sort****([**reverse=True**][,**key**=función])* ordena la lista. Si se emplea el parámetro **reverse** en orden descendente, si se usa **key**, con criterio de ordenamiento *función*
lista.***reverse()*** inierte el orden de la lista (primero va al último)
lista.***count****(valor)* cuenta la cantidad de apariciones de *valor* en la lista
lista.***index****(valor)* devuelve la posición de la primera aparición de *valor* en la lista

El código ASCII

sigla en inglés de American Standard Code for Information Interchange
(Código Estadounidense Estándar para el Intercambio de Información)

Caracteres ASCII de control

00	NULL	(carácter nulo)
01	SOH	(inicio encabezado)
02	STX	(inicio texto)
03	ETX	(fin de texto)
04	EOT	(fin transmisión)
05	ENQ	(consulta)
06	ACK	(reconocimiento)
07	BEL	(timbre)
08	BS	(retroceso)
09	HT	(tab horizontal)
10	LF	(nueva línea)
11	VT	(tab vertical)
12	FF	(nueva página)
13	CR	(retorno de carro)
14	SO	(desplaza afuera)
15	SI	(desplaza adentro)
16	DLE	(esc.vínculo datos)
17	DC1	(control disp. 1)
18	DC2	(control disp. 2)
19	DC3	(control disp. 3)
20	DC4	(control disp. 4)
21	NAK	(conf. negativa)
22	SYN	(inactividad sínc)
23	ETB	(fin bloque trans)
24	CAN	(cancelar)
25	EM	(fin del medio)
26	SUB	(sustitución)
27	ESC	(escape)
28	FS	(sep. archivos)
29	GS	(sep. grupos)
30	RS	(sep. registros)
31	US	(sep. unidades)
127	DEL	(suprimir)

Caracteres ASCII imprimibles

32	espacio	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

ASCII extendido (Página de código 437)

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp