**CURSOS DE PYTHON**

python es sensible a las mayusculas y minusculas

no es necesario declarar variables

# : se utiliza # para poner ocmentarios

pseudocodigo : es una notacion formalizada, concisoy puede ser leido

\ : cuando una linea termina con slash invertido significa que la linea continua en la linea siguiente

REGLAS PARA QUE UN PROGRAMA ESTE BIEN HECHO

**Alfabéticamente:** Un programa debe escribirse en una secuencia de comandos reconocible, por ejemplo, el Romano, Cirílico, etc.

* **Léxicamente:** Cada lenguaje de programación tiene su diccionario y necesitas dominarlo; afortunadamente, es mucho más simple y más pequeño que el diccionario de cualquier lenguaje natural.
* **Sintácticamente:**Cada idioma tiene sus reglas y deben ser obedecidas.
* **Semánticamente:**El programa tiene que tener sentido.

El lenguaje fue creado por Guido Van Rossum

**DSVENTAJAS DE PYTHON**

la ejecucion de un programa puede tomar arto tiempo

cython : convierte un codigo de python puro a c, para que se pueda ejecutar mas

rapido

jython : esto convirte a lenguaje java, no es compatible con python3

pypy : ayuda a la correccion de errores para cython

muchos de los componenetes de linus utiliza python

elementos de funciones de python:

causar algun efecto

evaluar un valor o algunos valores

argumentos

DEPURADORES PARA PYTHON

winpdb : este es un depurador

pdb : provee dos formas de depuracion

breakpoints : se pone puntos donde se detendra el programa

en el codigo ponemos como header

import pdb

y como punto de interrupcion ponemos

pdb.set\_trace()

post-motem : cuando se produce un error te permite revisar el codigo

al momento de correr el programa ponemos

python3 -m pdb nombre\_programa

nos manda el debugger

n : para pasar a la siguiente funcion

s : pasa de linea en linea

c : para correr el programa hasta el final

exit : para salir

list : nos muestra una porcion del codigo y nos dice en que linea estamos

break linea : nos pone un punto de interrupcion justo antes que se ejecute la linea puesta

break : poner solo break nos muestra la lista de puntos break y si estan habilitados o no

disable numero\_break : desactiva el break que pongamos sacado de la lista de break

enable numero\_break : vuelve a activar el punto de break

clear numero\_break : borra de la lista de breaks

ignore numero\_break numero\_veces: esta funcion hace que se ignore este break el numero de veces que se pone

**IDLE DE PYTHON**

idle-python3.6 : esta es la version de python que tenemos y nos lanza el idle de python

idle-python3.6 fichero: para abrir un fichero .py en el idle

**PROGRAMACION**

\n : es el salto de linea

**\ :** la barra invertida (escape)dentro de un cadena de caracteres significa que el siguiente carácter significa que el siguiente carácter es un carácter especial

n : viene de la palabra newline, osea que juntos hacen el comando de salto de linea

\\ : esto solo imprime una sola barra invertida ‘\’

\” : introduce comillas en el resultado de imprecion

si solo se pone “ le reconoce como fin o inicio de cadena

print(‘arreglo1’ , ‘ arreglo2’ , ‘ arreglo3’) : cuando se pone mas argumentos en print() este los concatena y pone espacio entre los argumentos

print(‘arreglo1’ , end=””) : el end=”” hace que se haga el salto de linea al final del print

print(‘arrreglo1’, variable, ‘arreglo2’,variable2) : asi se introduce variables al print

print(f”arreglo1 {variable} arreglo”) : es otra froma de introducir variables en print

print(‘arreglo1’, ‘arreglo2’ , ‘arreglo3’ , sep=’-’) : el comando sep=’-’ hace que al momento de imprimir y juntar los argumentos los separe con ‘-’ o con lo que se ponga ahi en lugar de un espacio

print(

"""

+==================================+

| Bienvenido a mi juego, muggle! |

| Introduce un número entero |

| y adivina qué número he |

| elegido para ti. |

| Entonces, |

| ¿Cuál es el número secreto? |

+==================================+

""")

poner 3 comillas hace que puedas usar el print asi

0o123 : el ‘0o’ (cero o) nos dice que el siguiente numero es un octal osea 123 equivale a 83

0x123 : el ‘0x’ (cero x) nos dice que el siguiente numero es un hexadecimal osea 123 equivale a 291

2\*\*3 : esto es igual a 2 a la potencia 3

cuando al menos uno de los operandos es flotante el resultado es flotante

6 / 3 : es una division

el resultado de una division es un flotante siempre, sin importar que la operación es entero

6 // 3 : esto es division entera

el rsultado es un numero entero, a la parte freccionaria simplemente la elimina

14%4 : esto es el **modulo, e**el modulo de esta operación es 2

14//4 =3

3\*4 =12

14-12 = 2 :esto es el modulo

9 % 6 % 2 : python hace esta operación de izquierda a derecha

9%6=3

3%2=1 este el resultado de python

2 \*\* 2 \*\* 3 : en potencia hace de derecha a izquierda

2\*\*3=8

2\*\*8=256 este es el resultado de python

x += 2 : esta operación hace que al valor de x le sume 2 y lo vuelva a guardar en x

-= : hace que se reste y lo guarda en la misma variable

\*= : hace que se multiplique y luego lo guarda en el variable

/= : hace que se divida y luego lo guarda en la variable

sepuede hacer eso con cualquier operador

round(valor , numero\_decimales) : esta funcion redondea el valor que se le pasa al numero de decimales que se especifica

input(‘dime algo’) : imprime elmensaje y pone el terminal en modo escucha

siempre entrega una cadena

type(variable) : te dice que tipo de dato es la variable puesta

max(var1,var2,var3): esta funcion entrega el con valor maximo

min(var1,var2,var3): esta funcion entrega el valor minimo

range(numero) : la funcion genera un conjunto de numero consecutivos desde el 0 – (numero-1)

numero tiene que ser entero

este comando genera la cantidad de numeros que dice numero desde 0 (cero) por eso solo va a (numero -1)

range(2,8) : este genra numeros desde el 2 hasta el 7

el ultimo valor se descarta

range(2,8,3) : el tercer valor dice el salto que dara, es decir que saltara de 3 en 3

el rango contara el 2,5 : porque el siguiente numero seria el 8 pero no entra en el rango

CASTEAR VARIABLES

int(string) : convierte un string a introduce

float(string) : convierte un string a flotante

str(int) : convierte un numero a string

cadena.upper() : el .upper() vuelve a todo lo que este en cadena a mayuscualas

CONCATENAR

string1 + string2 : el signo mas cuando se lo usa entre dos cadenas, este las concatena

OPERADORES COMPARATIVOS

a == b : pregunta si ‘a’ es igual a ‘b’

1. == 1 : en el caso de comparar un entero con un real, python puede volver el entero a real

la respuesta es True

a != b : pregunta si ‘a’ **no**  es igual a ‘b’

si son distintos devuelve True

a > b : pregunta si ‘a’ es mayor que ‘b’

a >= b : pregunta si ‘a’ es mayor o igual que ‘b’

a and b : es el comando de conjuncion

a or b : es el comando de disyuncion

not a : es la negacion de a

not (a and b) == (not a) or (not b)

not (a or b) == (not a) and (not b)

OPERADORES BITWISE

& : es and a nivel de bits

a = a & b

a &= b

| : es or a nivel de bits

a = a | b

a |= b

~ : negacion a nivel de bits

^ : es xor a nivel de bits

a = a ^ b

a ^= b

0 0 = 0

0 1 = 1

1 0 = 1

1 1 = 0

variable >> n\_bits : desplaza a la derechalos bits de variable n\_bits puestos

variable << n\_bits : desplaza a la izquierda los bits de variable n\_bits puestos

**ARRAY DE VALORES(LISTAS)**

lista = [] : creas lal ista vacia

lista =[1,2,3,4,5] : declaras la variable de 5 valores

fila = [i for i in range(4)] #creamos una lista con valores del 0 al 3

print(fila)

columna=[j for j in fila if j%2==0] #creamos otra lista solo con los valores inpares de la lista fila

print(columna)

tablero = [[EMPTY for i in range(8)] for j in range(8)]

EMPTY : es una variable que creamos antes que esta vacia

EMPTY=’’

estamos creando una matriz de 8x8

la lista pueden recibir todo tipos de datos simultaneamente

se peude poner otra lista en **una posicion de la lista**

lista = lista2\*2

cuadno multiplicamos una lista por un numero devuelve la misma lista repetido el numero de veces que pusimos

lista1=lista2 : copias el contenido de una lista a otra lsita

los dos nombres apuntan al mismo espacio de memoria

cuando cambiamos algo en cualquiera de las listas, tambien se cambia en la otra

del lista2 : este comando solo borra el nombre lista2

el contenido sigue vivo en lista1

lista1 = lista2[:] : poner los dos puntos hace que se copie solo el contenida

asi tenemos 2 listas independientes

lista1 = lista2[2:4] : esto copia los valores de la poscion 2 hasta la **posicion 3**

si se omite el inicio o fin dentro de los corchetes, se toma que el inicio o fin de la lista

los indices empiezan desde el 0 si vemos desde la izquierda

los indices empiezan desde el -1 y sigue -2 si vemos desde la derecha

variable[0]=3

cambiar de valor alguna posicion se llama indexion

las posiciones del arreglo empieza desde el 0(cero)

**los indices que se ponen dentro del corchete son de tipo str, no son numeros**

len(arreglo) : nos dice la longitud del arreglo

del areglo[posicion] : del elimina la posicion q se espesifique del arreglo

del lista[:] : elimina el contenido de la lista

deja la lista vacia

lista.append(valor) : .append aumenta valor al final de la lista

lista.insert(posicion,valor) : .insert inserta el valor a posicion de la lista

los valores que ya estan el la lista recorren hacia la derecha desde la posicion especificada

lista.sort() : ordena la lista y lo guarda en la lista misma

tambien ordena letras

sorted(lista) : **entrega la lista ordena**

es necesario guardarlo enotra variable

lista.reverse() : invierte la lista

lista.count(valor) : .count() **devuelve** el numero de veces que se repite el valor en lista

este metodo solo sirve con listas y tuplas

lista = list(secuencia) : list() **devuelve** la secuencia convertida en lista

si se le pasa un diccionario como secuencia la convierte en lista

pone **solo las claves** en la lista

BUSCAR EN LAS LISTAS

valor in lista : el in pregunta si el valor esta dentro de la lista

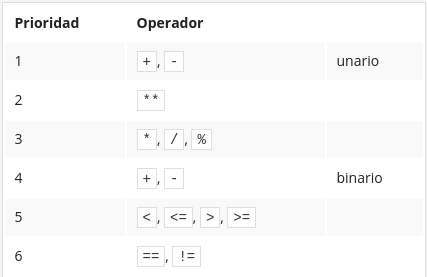
si el valor esta devuelve True

valor not in lista : el not in pregunta si el valor esta ausente en la lista

si el valor **estas ausente**  en la lista devuelve True

****

**PRIORIDADES DE OPERADORES**

****

**CONDICION IF**

if condicion :

funcion a ejecutar1()

funcion a ejecutar2()

elif:

funcion a ejecutar3()

funcion a ejecutar4()

else:

funcion a ejecutar5()

funcion a ejecutar6()

funcion7()

para las funciones que se ejecutaran dentro del if poner todas al mismo nivel de tab o espacios

funcion3() : ya no esta dentro del if por que no esta al mismo nivel de tab o espacios que las funciones dentro del if

anidamiento: poner un if dentro de otro if

cascada : cuando usamos elif

**CICLO WHILE**

while condicion:

instrucciones

else:

instrucciones

cuerpo del ciclo : son las instrucciones ejecutads dentro del while

while puede tener else que se ejecuta cuando la condicion no se cumple

la parte else solo se ejecuta una vez

**CICLO FOR**

for i in range (100):

comandos a ejecutar()

pass

else:

comando a ejecutar()

pass : esta funcion no hace absolutamente nada

la sintaxis del ciclo for exije si o si al menos una funcion

else : las intrucciones dentro de else solo se ejecutan cuando la variable de control este fuera del rango

si termina la cuenta la variable de control se queda con el ultimo valor

DULSES SINTACTICOS

break : sale del cilo inmediatamente, continua con el codigo

continue : pasa al final del cuerpo del ciclo, e inicia el siguiente ciclo

**MODULOS**

import time : importamos el moduo time

time.sleep(1) : esta funcion hace un sleep de 1 segundo

**METODOS**

resultado = data.metodo(argumento)

obtiene datos, puede crear datos y por lo general produce datos

es propiedad de los datos para quien trabaja

**FUNCIONES**

tipos de funciones:

funciones integradas: sonlas que vienen con python

print(), input()

de los modulos preinstalados : para invocarlas se eguin pasos extras

funciones directamente del codigo: estan escritas dentro del programa que se esta ejecutando

de clases : se vera mas adelante

la definicion de las funciones deben estar antes de la invocacion

cuando se le manda argumentos a una funcion al momento de invocarla y esta trabaja con argumentos, nos lanzara error

los parametros dentro de una funcion solo toman valor dentro de la funcion

se puede tener variables con el mismo nombre que los parametros de una funcion, no se afectaran entre si

si no se completa los parametros requeridos por la funcion al momento de invocarlo da error

una funcion puede modificar una variable del programa pero este valor solo se mantiene dentro la fucnion

al salir de la funcion la variable del programa recupera el valor del programa

global variable : al declarar una variable global dentro de una funcion se la roconoce en todo el programación

DEFINIR FUNCION

def nombre\_funcion(parametro1, parametro2=valor2):

cuerpo de la funcion

return valor

return

nombre \_funcion()

asi se la define a la funcion

al momento de definir la funcion se puede dejar un valor por defecto a algun argumento , y asi se omite el mismo al momento de invocar la funcion

si el argumento tiene un valor por defecto pero al mento de invocarlo se le asigna un nuevo valor, este argumento tomara el valor nuevo

poner las variables con valores predefinidas al final

cuando se pone la palabra reservada return sin nada despues hace que termine la funcion

se asemeja a break en los ciclos

cuando no se pone nada para devolver a la funcion, esta devuelve **None**

None: es una palabra reservada que expresa vacio

se usa cuando se asigna una variable

cuando se compara con una variable

cuando se pone una variable o un valor a lado de return, se entrega esta como resultado al momento de invocar la fucion

una funcion puede invocar otras funciones

recursividad : se puede invocar a la funcion dentro de la misma funcion

ejemplo: /curso\_python/programas/codigo15.py

desventaja de la recursividad es que ocnsume bastante memoria

**tener cuidado que la recursividad sea infinita**

**poner limites o condiciones**

INVOCAR FUNCION

resultado = funcion(parametro1, parametro2)

asi se la invoca y encaso de devuelva algun valor se la guarda en variable

se puede mandar una lista como parametro de la funcion, siempre que la funcion sepa manejarla

para mandar los parametros al momento de invocar al funcion se puede hacer de dos formas:

funcion(valor1, valor2)

poner lo valores de los parametros de los parametros en el orden que se definio tomando en cuenta su tipo y que corresponde a cada parametro de la funcion

funcion(parametro2=valor 2 , parametro1=valor1)

se puede poner en desorden pero especificando el nombre clave del parametro al cual se le esta asignando las variables

**TIPO DE SECUENCIA**

\*MUTABLIDAD : es la habilidad de un dato de cambiar su valor en el transcurso del programa

inmutablidad: cuadno el valor del dato no puede cambiar

in situ : cuando se cambia un valor de un dato

es un tipo de valor que puede almacenar mas de un valor o ninguno

los cuales pueden ser examinados secuencialmente por el bucle for

**tipos de secuencias**

lista[] : es una secuencia mutable

tupla[] : es un asecuencia inmutable

se puede leer una tupla pero no modificarla

tupla[0] :para leer la tupla en posicion 0

tupla = tupla1 + tupla2 : se puede unir tuplas

pero solo entre tuplas, no con una lista

tupla = (1, 2, 3, 4 ,5)

tupla = 1, 2, 3, 4, 5

tupla = ()

tupla = (1 ,)

la tuplas se lapuede crar de estas dos maneras

poniendo entre parantesis o simplemente separando por coma(,)

para una tupla de un valor **poner el valor y una coma (,)**

las tuplas pueden tener cualquier tipo de valor dentro

a, b, c = tupla : los valores dentro de tupla se guardan en a,b,c cada uno

ESCEPCIONES DE TUPLAS

tupla \*= 3 : repetimos el contenido de la tupla 3 veces y la guardamos en ella misma

tupla += (valor, ) : aumentamos el valor a la tupla

del tupla : se peude eliminar una tupla entera

tupla1 = tupla2 : tupla1 va a ser igual a tupla2 sin importar que habia antes en tupla1

tupla1 = sorted(tupla2) : parapoder dordenar una tupla se puede usar sorted()

es necesario guardarlo en otra tupla

lista = list(tupla)

el comando list() **devuelve** la tupla convertida en lista

este comando vuevlve todo los iterables en lista

en caso de pasarle una cadena, pone cada carácter en un indice

si se le pasa un rango convierte este en lista

pone cada valor del rango en un indice

tupla = tuple(lista)

el comando tuple() **devuelve** la lista convertida en tupla

este comando vuelve todo los iterables en tupla

si se le pasa una cadena, pone cada carácter en cada indice

si se le pasa un rango, lo convierte en tupla

pone cada carácter en un indice

si se le pasa un diccionario **convierte las claves en tupla**

pone solo las claves del diccionario en la tupla

**DICCIONARIO** (tipo de dato)

es un tipo de dato mutable

es un conjunto de palabras clave y valores

los pares se encierran **en llaves**

los diccionarios no mantienen el orden

el orden varia dependiendo la version de python que se tenga

al imprimir el diccionario se ve el orden guardado

diccionario = { ‘clave1’ : ‘valor1’ , ‘clave2’ : ‘valor2’ , ‘clave3’ : ‘valor3’ }

variable = diccionario[‘clave1’] #obtener el valor de la clave1

variable = diccionario.get(‘clave1’ , ‘valor\_faul’)

.get() : devuelve el valor de la clave1 y en caso de que la clave1 no este en el diccionario devuelve el valor\_faul

diccionario[‘clave\_nueva’] = valor\_nuevo #aumentar nueva clave con su valor

diccionario.update({‘clave\_nueva’ : ‘valor\_nuevo’})

.update(): tambien sirve para insertar nuevo par al diccionarios

del diccionario[‘clave’] #eliminar un par del diccionarios

solo necesitamos poner la clave y tambien se borrara el valor

diccionario.popitem() : elimina el ultimo par del diccionario

del orden en el que se escribio en el codigo

diccionario.clear() : borra todo lo que esta dentro del diccionario

a, b, c = adiccionario : se guardan **las claves** del diccionario en las variable a,b,c

clave in diccionario : in pregunta si la clave esta en el diccionarios

devuelve True si la clave esta

devuelve False si la clave no esta

calve not in diccionario : not in pregunta **si la clave NO esta** en el diccionario

devuelve True si la clave **NO esta**

devuelve False si la clave **SI esta**

para obtener el valor1 tenemos que preguntar por la clave1

las claves y los valores pueden ser de cualquier tipo

cada clave debe de ser unica

len(diccionario) : nos devuelve la cantidad de paras que tiene (clave y valor)

for clave in diccionario.keys() :

#for clave in sorted(diccionario):

print(clave , ‘ → ’ , diccionario[clave])

diccionario.keys() : el .keys() nos devuelve una lista con **las palabras clave** del diccionario

para poder usar el ciclo for se tiene que poner el .keys() para tratarlo como una lista

diccionario.values() : el .values() **nos devuelve**  una lista con **los valores** del diccionario

diccionario\_copia = diccionario.copy() : el .copy() **entrega** una copia del diccionario

creas una lista independiente, los cambios en el original no afecta la copia

sorted(diccionario) : podemos usar esto en lugar de ‘diccionario.keys()’

este comando nos devuelve una lista con las claves del diccionario **y ademas ordenados**

CASTEO DE DICCIONARIO

#la lista/tupla tiene que estar en este formato

colores = (("verde", "#008000"), ("azul", "#0000FF"))

colDict = dict(colores)

print(colDict)

dict() : es la funcion que convierte las listas o tuplas en diccionarios

necesita tener formato

DICCIONARIO.ITEMS()

esto nos devuelve un objeto de vista

es un iterable

nos devuelve **una lista de pares de tuplas donde tiene los pares (clave , valor) del diccionarios**

cuando modificamos el diccionario, el objeto de vista obtenido obtenida con el comadno .items() tambien se actualiza

for clave,valor in diccionario.items():

print(clave , ‘ => ‘ , valor)

a los valores de este objeto solo se puede acceder con el ciclo formaliza