RELATORIA 1

Alejandro Patiño

September 2022

1 OBJETIVOS

1. espero aprender a utilizar python de forma eficiente, entender sus características y aprender a trabajar con ellas
2. saber como trabajar con cada uno de los diferentes tipos de datos
3. aprender a utilizar los diferentes operadores y aplicarlos en un programa
4. aprender a utilizar las funciones integradas que nos deja python y sus aplicaciones en una lógica computacional
5. saber usar el condicional if y como aplicarlo
6. aprender a aplicar cambios de variable en memoria para optimizar el programa

2 23-08-2022

en esta clase se configuró el entorno de trabajo utilizando las siguientes herramientas: 1. introducción al lenguaje Python

2. instalación visual estudio code

   se basó en seguir una instalación guiada por el profesor quien nos dio un enlace de instalación

3. instalación git

   se basó en seguir una instalación guiada por el profesor quien nos dio un enlace de instalación

4. cuenta github

   buscamos github.com y se creó una cuenta con la cual posteriormente se siguió al profesor para seguir con las clases

el curso se basara en el uso de lenguaje de programación Python para lo cual utilizaremos como herramienta de escritura de código al programa visual estudio code. además, se realizara la instalación de github con el objeto de subir los repositorios a la nube

3 25-08-2022

Aprenderemos a hacer uso de GitHub, las cuales son herramientas para el uso y manejo de repositorios locales y en la nube.

pasos para crear repositorios =>

1. tener Git y cuenta GitHub
2. configurar usuario y correo abrimos el terminal y escribimos lo SIGUIENTES git config –global user.name “nombre de usuario” git config –global user.email “correo con el que se registró”
3. ir al icono (3 nodos)
4. ir al working directory (inicializar un repositorio) dando click al botón inicializar repositorio
5. llevarlo al stage area: se selecciona el símbolo “+” que se encuentra sobre el nombre del archivo
6. llevarlo al commit area: primero se hace un mensaje descriptivo para ello se selecciona el botón “commit”

7. llevarlo a la nube. para ello selecciono el icono en forma de nube
8. añadir nuevos documentos a la nube. cuando se vaya a añadir información, repetimos los pasos desde el 5
9. sincronización con la nube al terminar se pedirá sincronizar los datos, oprimimos sincronize y aceptamos las ventanas emergentes

veremos los fundamentos de python a continuación

4 25-08-2022

4.1 Características de Python

- lenguaje de alto nivel (esto lo hace mas fácil al entendimiento humano
- interpretado (traduce del lenguaje de programación a lenguaje maquina)
- paradigma funcional y orientado a objetos
- tipado dinámico

4.2 Tipos de Datos

- Booleanos: True, False (son datos que solo toman dos valores Verdadero y Falso, 1 o 0 , etc. . .)

  algo a tener en cuenta en los datos booleanos es que los números, sean enteros o flotantes. los strings y las listas, tuplas y conjuntos tienen un valor booleano aplicado. esto lo veremos más adelante .

- enteros: -10, 0, 10, 9999 (son todos los valores positivos o negativos sin decimales)

  estos los podremos utilizar con una cantidad de memoria para reducir la carga de información y operaciones al computador esto generalmente lo haremos cuando estemos trabajando con volúmenes de datos muy grandes. más adelante veremos como .

- flotantes: 12.5, 15.46, 100.0 (son todos los valores con decimales, tienen una forma de memoria diferente)

  al igual que los enteros podremos trabajar con más memoria que a la vez serán más decimales, y por tanto, más precisión en diferentes ejercicios. el cambio de decimales y cambio mayor o menor memoria se verá mas adelante

- strings: “”, ’ ’,”123”, “ “ “”,”carros” (son todas las variables que se tome en texto)

  con estos datos debemos tener muy en cuenta poner las comillas

- listas: [], [1,2,3], [“a”,“b”,“c”], [1,2,“a”,“b”]. (las listas son mutables, se pueden cambiar)

  las listas son muy importantes ya que nos ayudan a manejar múltiples datos de forma ordenada y aplicarlas, bajo operaciones estadísticas u operaciones aritméticas, para la solución de problemas en los que necesitemos más de una variable

- tuplas: (), (1,2,3),(“a”,“b”,“c”), (1,2,“a”,“b”). (las tuplas son inmutables)

  deberemos tomar en cuenta que las tuplas no se podrán cambiar y por esto será a veces conveniente cambiarlas a listas bajo operaciones que veremos a continuación

- diccionarios:{clave1:valor, clave2:valor, clave3:valor} (son necesarios para agrupar funciones)
- conjuntos:{1,2,3,4,5}, {“a”,“b”,“c”} (sirven en la demostración matemática de algunas hipótesis)

algunos ejercicios prácticos para mejorar

[ ]:
```
# crear variables y asignarles los siguientes
#Tipos de Datos
```

```
#Enteros: 1,2,3,999
# luego reste sucesivamente del ultimo al primero y almacenarlo en una variable␣
 ↪llamada resultado1

from pickletools import string1  #esto lo hacemos para importar la funcion string1 de␣
 ↪la libreria pickletools

a=1;
b=2;
c=3;
d=999;
resultado1= d-c-b-a;
print("el resultado es: ",resultado1)

#Flotantes: 15.2, 29.5, 18.28
#luego divida sucesivamente del primero al ultimo y almacenelo en una variable llamada␣
 ↪resultado2
a=15.2;
b=29.5;
c=18.28;
resultado2=((a/b)/c);
print("el resultado es: ",resultado2)

#Strings: "123", "Cristian"
#luego sume ambas variables y determine si la operacion es posible, si asi es,␣
 ↪almacenelo en una variable de
#su eleccion
a="123"
b="Cristian"
resultado3=a+b
print(resultado3)
#en este caso no es una suma como tal, es concatenación


######## para pensar ########

#busque una manera de convertir:
# un entero a un flotante
y=85;
print(float(y))
# pasamos de una variable tipo entero a una flotante con la funcion float() que␣
 ↪convierte en flotante


# un flotante a un entero
x=2.6;
print(int(x))
# pasamos de una variable flotante a un entero con la funcion int() que toma la parte␣
 ↪entera del numero dado


# un string a un entero y flotante
a="56"
```

```
b="20.8"
c=int(a)+float(b)
print("el resultado de esta suma es ",c)
#esto lo logramos aplicando la funcion int() y float() que "ven" dentro de las␣
 ↪variables string
#la funcion int() para enteros y la funcion float() para flotantes


# un numero a un string
z=65
print(str(z))
#esto lo logramos gracias a la función str() que transforma una variable tipo int o␣
 ↪float a string
```

5 30/08/2022

5.1 OPERADORES

1. operador de asignación: =
2. operadores aritméticos: + (suma) ,- (resta) ,* (multiplicación) ,/ (división) ,// (división entera) ,% (modulo) y ** (potenciación)

   se debe tener en cuenta para que son cada uno ya que es fácil equivocarse. más adelante veremos algunas propiedades que tiene la división entera, el modulo y la potenciación. que nos serán útiles más adelante
3. operadores lógicos: and (conjunción) , or (disyunción) , not (negación)

   debemos recordar que los operadores lógicos solo dejan salir datos booleanos, salvo en algunos casos que veremos más adelante con los datos tipo string
4. operadores de comparación(se puede utilizar con booleanos, string ,números en general): > (mayor que) , >= (mayor o igual que), < (menor que) , <= (menor o igual que) , != (diferente de) y == (igual a)

   vale aclarar que no es conveniente el confundir == con =, ya que == es un comparador y el operador = es un asignador
5. operadores de pertenencia: in (pertenece a) , not in (no pertenece a)

   estos operadores se usan generalmente con elementos y conjuntos. sirve en general para seleccionar algún elemento de un conjunto
6. operadores de conjuntos: |(union), &(interseccion), -(diferencia)

   son operaciones entre conjuntos, muchas veces son necesarias en teoría de números o en aplicaciones matemáticas con aplicando computadores

algunos ejercicios prácticos para ampliar conocimientos

[ ]:
```
# operadores de asignacion


a=1
b=3
c=5
```

```
#operadores aritmeticos


#realice las siguientes operaciones mentalmente

print(3+9)
print(3.0+9)
print(9**0.5)
print(2**32)
print(19//2)
print(19%3)
print(9/3)
print("hola "+"mundo")
print("hola " * 3)
print(["A"]+[1,2,3])
print([]+[])
print((1,2,3)+(1,))

#se debe tener mucho cuidado para concatenar tuplas ese (1,) es una tupla, pero
#el (1) no es una tupla, es solo una agrupacion de un 1 por tanto un numero, entero en
 ↪este caso

print("_______________________________")

# esto lo hacemos para separar cada grupo de ejercicios de ejercicios

#operadores logicos


#realice las operaciones mentalmente

print(True and True)
print(False and True)
print(False and False)
print(not True)
print(not False)
print(True or True)
print(False or True)
print(1 and 1)
print(0 and 1)
print(1 and 3)
#todos los numeros enteros son positivos, tomando 0 como Falso y 1 como Verdadero

#todos los strings son verdaderos a excepcion del "" (string vacio)""

print(1 and "hola")
print(0 and 3)
print(0 and "hola")

# estas operaciones logicas se pueden tomar como un si, entonces . esto es muy
 ↪importante

print("hola" or "verdadero")
```

```
print(1 or 3)
print("verdadero" or "hola")

# de esta forma vemos que con ese operador or nos devolverá el primer valor


print("_____________________________")


# Operadores de comparacion


# debemos tener en cuenta que deben devolver booleanos, independientemente del tipo de␣
 ↪variable

#realizar las siguientes operaciones mentalmente

print(1 > 2)
print(1 < 3)
print(1 == 1) #debemos poner == para hacer comparaciones, el otro = es de asignacion
print(2 != 1)
print(3 <= 3)
print(5 >= 2)
print(4 > True)
print(True > False)

#no se puede comparar string con booleanos ("hola" > True)

print([] > [1,2,3]) # recordemos que [] se toma como falso (por lo tanto 0) y␣
 ↪cualquier lista es verdadero
#(por lo tanto 1)

print("%" > "a")  # esto lo tomamos segun el codigo ASCII y los comparamos en los␣
 ↪string

print("________________________")

# operadores de pertenencia


#estos operadores solo responden en booleano

#realizar las siguientes operaciones

print("a" in "abcdefg")
print("A" in "HOLA MUNDOOOO")
print(1 in [1,2,3])
print(1 in ["1","2","3"])
print("hola" not in "hola mundo como vais")
```

6 1/09/2022

6.1 Funciones Integradas

son funciones predefinidas en Python (Funciones built)

- Entrada y Salida: input() (esto lo usamos para que el usuario ingrese un dato, el dato ingresado entra formato string) , print() (lo usamos para mostrar en pantalla lo que esta dentro de los paréntesis), format() (lo usamos para poner en un formato, puede ser formato científico, en alguna base , etc. . .)
- Ayuda: help() (lo usamos para saber con que se utiliza una función en especifico) , dir() (lo usamos para saber las utilidades de una función), type() (es para saber de que tipo es la variable dentro de los paréntesis)
- Matemáticas: abs() (esto lo usamos para sacar el valor absoluto de un cierto numero, flotante o entero), round() (esto lo usamos para redondear un valor con un cierto numero de decimales), pow(,) (que devuelve el resultado de elevar x a la y siempre devuelve un numero real), . . .
- Conversiones: int() (pasa cualquier tipo de variable a entero), float() (pasa cualquier tipo de variable a flotante), str() (cambia cualquier tipo de variable a string), complex() (cambia cualquier variable numérica a compleja), bool() (convierte una variable, la que sea en un booleano), set() (cambia una variable a conjuntos), list() (cambia una variable a lista, generalmente lo utilizamos con variables tipo string), tuple() (convierte la variable a una tupla), bin() (cambia una variable a base binaria), oct()(cambia una variable a base octal), hex() (cambia una variable a base hexadecimal), int () (esta también sirve para cambiar a base decimal)
- secuencias: range() (se utiliza para representar una secuencia inmutable de números), enumerate() (enumera los elementos de una lista, conjunto o tupla), zip() (acepta objetos iterables como listas, cadenas y tuplas como argumentos y devuelve un solo iterable)
- Operaciones en secuencias: len() (mide la cantidad de elementos que tiene una secuencia) , sum() (suma todos los elementos de una secuencia) , max() (mira el valor máximo que tiene una secuencia), min() (mira el valor que tiene una secuencia), sorted() (crea una nueva lista ordenada a partir de un iterable), map() (aplicar una función a cada elemento en un iterable (como una lista o un diccionario) y devolver un nuevo iterado para recuperar los resultados), filter() (crear un nuevo iterado a partir de un iterable existente (como una lista o un diccionario) que filtrará de forma eficiente los elementos usando una función que proporcionamos)

estas funciones son las básicas y ya integradas en Python, sus algoritmos ya están optimizados y podemos realizar múltiples cálculos y formateos

Nota: para la Operación Format() debemos tomar la clase separado de “,” y ponemos entre comillas “ne” para formato científico, “nf” para formato flotante. donde “n” es el numero que queramos de decimales

[ ]:
```
                #funciones entrada/salida

            # Funcion input/print

a=input("ingrse su nombre: ")
print("hola ", a , "estas muy guapo hoy")


#solicite la edad y muestre si es mayor de edad o no es mayor de edad

edad=int(input("por favor dame tu edad: "))
if edad>=18:
    print("uy ya eres legal ;) ")
elif edad<18:
```

```
    print("tu eres menor de edad ")

# solicite una clave de entrada y muestre en pantalla si es correcta o incorrecta
#clave=9876
#
cl_Original=9876
cl_entrada=int(input("ingrse su clave, por favor: "))
if cl_Original == cl_entrada:
    print("tu clave es correcta, no se te olvido esta vez")
else :
    print("tu clave es falsa, anotesela en un papelito mejor")


         # Funcion Format

numero=193.8576
fomat_Cientifico= format( numero, "e")
print("formato scientifico : ", fomat_Cientifico)
fomat_Cientifico= format( numero, "2e")
print("formato scientifico con dos decimales : ", fomat_Cientifico)
fomat_Cientifico= format( numero, "6e")
print("formato scientifico con 6 decimales : ", fomat_Cientifico)

numero= 12.856723
formato_Flotante= format(numero,"3f")
print("formato flotante con 3 decimas: ", formato_Flotante)

cadena= "hola mundo"
formato_centrado= format(cadena, "^10")    # para que el texto quede centrado
formato_derecha= format(cadena,">10")      # para que el texto quede a la derecha
formato_izquierda= format(cadena,"<10")    # para que el texto quede a la izquierda

print("formato centrado  : ", formato_centrado)
print("formato derecha  : ", formato_derecha)
print("formato izquierda  : ", formato_izquierda)



                  # Funciones de conversión

# convertir a binario, octal y hexadecimal

decimal=56

convBinario= bin(decimal)
convOctal= oct(decimal)
convHex= hex(decimal)

print(" aparecen bin, oct, hexadecimal", convBinario, convOctal, convHex)

          # Funciones de ayuda (dir)

cadena= "hola mundoo"
```

```
lista=[1,2,3]
entero= 19

print("funcionalidades para cadena ==> /n/n", dir(cadena) )
print("funcionalidades para lista ==> /n/n",  dir(lista ) )
print("funcionalidades para entero ==> /n/n", dir( entero) )




         # Funciones para secuencias

secuencia= range(1,11,1) # range(inicio,final,salto) el final no lo toma
# cuando el salto es 1 no se necesita el tercer parametro. range no se puede aplicar
 ↪print

print(list(secuencia))   #por lo tanto tenemos que pasarlo a lista

# numeros de -10 al 10 con salto 2

print(list(range(-10,11,2)))

# numeros multiplos de 3 desde el -10 hasta el 5
print(list(range(-9,5,3)))

# numeros del 10 al 0
print(list(range(10,-1,-1)))

# numeros multiplos de 3 y 5 del 1 al 1000 al reves

print(list(range(990,0,-15)))


# para ver el tamaño de una secuencia lo hacemos con len()
# para sumar todos los componentes de una secuencia lo hacemos con sum()
# aplicamos el minimo y el maximo con min() y max() respectivamente
# lo revertimos con reversed()

secuencia = range(1,200,4)
lista= [1,2,5,0,9,7]

print("tamaño secuencia ==>", len(secuencia))
print("tamaño lista==>", len(lista))

print("el maximo de la secuencia es ", max(secuencia))
print("el maximo de la lista es ", max(lista))

print("el minimo de la secuencia es ", min(secuencia))
print("el minimo de la lista es ", min(lista))

print("revertir la secuencia ", list(reversed(secuencia)))
print("revertir la lista ", list(reversed(lista)))
```

```
# repetir el ejercicio anterior usando reversed

print(reversed(list(range(1,990,15))))
```

7 06 - 09 - 2022

7.1 CONDICIONAL IF

Cumple con la operación lógica si=> entonces. Si la condición es verdadera => Las sentencias se ejecutan. En caso contrario (condición es Falsa) => Las sentencias se ignoran.

La notación utilizada en python es como sigue:

if : ....

if : elif <condición 2>: elif <condición 3>: elif <condición 4>: else:

[ ]:
```
# Pida a un usuario su nombre y su edad. Determine si es mayor de edad,
# y muestre un mensaje en pantalla diciendo:
# <NOMBRE>, usted es mayor/menor de edad


nombre = input("Ingrese su nombre: ")
edad = int(input("Ingrese la edad: "))

if 18<=edad<=150:
    print("{}, usted es mayor de edad".format(nombre))
elif 0<edad<18:
    print("{} usted es menor de edad".format(nombre))

"""
Realice un programa que calcule
el mayor de tres números
"""
a = 3
b = 5
c = 5
if a>=b and a>=c:
   print("{} es el numero mayor".format(a))
elif b>=a and b>=c:
    print("{} es el numero mayor".format(b))
elif c>=a and c>=b:
    print("{} es el numero mayor".format(c))

"""
*salario base = 1 000 000
Realice un programa que
calcule el salario de un vendedor
de seguros,teniendo en cuenta
las siguientes condiciones =>

=> ventas => entre [5, 20] seguros =>
              aumento del 20% sobre la base
```

```
=> ventas => entre [21, 50] seguros =>
             aumento del 30% sobre la base
=> ventas =>  entre [51, infinito] seguros =>
             aumento del 35% sobre la base

"""

salario_base = 1_000_000
ventas = int(input("Numero de ventas: "))
salario_total = 0
condicion1 = (5 <= ventas <= 20)
condicion2 = (21 <= ventas <= 50)
condicion3 = (ventas >= 51)

if condicion1:
    salario_total = salario_base + 0.2 * salario_base
elif condicion2:
    salario_total = salario_base + 0.3 * salario_base
elif condicion3:
    salario_total = salario_base + 0.35 * salario_base
print("El salario total es {}".format(salario_total))


"""
Una contraseña de un programa, debe incluir:

* Contenga mayusculas
* Contenga minusculas
* Contenga números
* Caracteres especiales
* Por lo menos 8 caracteres en total

Determine si al ingresar una contraseña, esta cumple con todas las
anteriores condiciones. """

#contraseña = "jsfdlkLJKJ5678/8%$"
#validez = False
#
#condicion1 = list(contraseña) in␣
↪[a,b,c,d,e,f,g,h,i,j,k,l,m,n,ñ,o,p,q,r,s,t,u,v,w,x,y,z]
#condicion2 = list(contraseña) in [A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z]
#condicion3 = list(contraseña) in [0,1,2,3,4,5,6,7,8,9]
#condicion4 = <Caracteres especiales>
#condicion5 = <Por lo menos 8 caracteres en total>
#
#if (condicion1 and condicion2 and condicion3 and condicion4 and condicion5):
#    print("validez = True")
#else:
#    print("validez = False")

contraseña= input("por favor ingrese su contraseña")
tamaño= len(contraseña)
cont=0
```

```
validez= False

while cont< tamaño:
    if contraseña[cont] in "qwertyuiopasdfghjklñzxcvbnm":
        validez= True
    if contraseña[cont] in "QWERYUIOPASDFGHJKLÑZXCVBNM":
        validez= True
    if contraseña[cont] in "1234567890":
        validez= True
    if contraseña[cont] in ",;.:-_¨`^+*!Çç'¡?¡¿!˜¬#@ºª\·$%&/()":
        validez= True
    if len(contraseña)<=8 :
        validez=True
print(validez)
```

8 CURSOS EXTRAS

8.1 INTRODUCCIÓN A LAS CIENCIAS DE LA COMPUTACIÓN

en este curso se verán Nociones del concepto de Computación, Procesos algorítmicos, Sistemas de computo, Elementos del pensamiento computacional: descomposición, reconocimiento de patrones, abstracción, y algoritmos. Métodos de Computación, Dividir y Conquistar, Recursión, Reintento (bactracking). Descripción de algoritmos en seudo código, Tipos de datos, Estructuras de control, Estructuras básicas de datos e Implementación de algoritmos en un lenguaje de programación procedimental (aplicando Python)

Nota: se recomienda que para hacer un ejercicio de programación, primero se enumeran los pasos en forma matemática y se aplican cada paso en forma computacional

Proceso de solución computacional de un problema

1. análisis del problema: tratar de dar a entender el problema en todas sus partes, ojalá en forma matemática para más facilidad
2. diseño de la solución: analizar los datos que se dan y estructurarlos en un algoritmo lógico
3. implementación: representar en un lenguaje de programación e implementarlo
4. prueba y depuración: probar y corregir el programa de forma lógica

Division Entera dados A y B números. el cociente q de la división entera de A entre B esta dado por q=A//B. y es igual al numero de veces que se puede restar B al numero A antes de que sea negativo el resultado. otra forma es tomar la división de A entre B sin tomar números decimales tomando r= residuo que es un numero entre 0 y b. el cual se halla r= A%B. tenemos una propiedad muy importante A= (q*B)+r debemos tomar en cuenta que B tiene que ser distinto de 0

8.1.1 Eficiencia Computacional

computacionalmente, es mas eficiente dividir o multiplicar por 2 y hacer sumas, porque el computador al ser un sistema que trabaja en binario entonces le toma mucho menos tiempo la traducción. también debemos tener en cuenta que es mucho mas eficiente trabajar con un lenguaje basado en 0 y 1 que en un lenguaje traducido, por lo menos computacionalmente, ya que generalmente es mas complicado el lenguaje maquina que el lenguaje traducido para los humanos.

el saber si es par o impar es muy eficiente computacionalemente, por lo tanto es bueno tenerlo en cuenta para los algoritmos

8.1.2 Modulo

a es multiplo de b si y solo si existe un numero k tal que a=k*b o a % abs(b)=0. esto lo hacemos ya que el modulo esta definido para b>0

8.1.3 Cambios de variable en memoria

cuando se está manejando una cantidad muy grande de datos, tenemos que pensar en la memoria ya que esta no es infinita, se colma muy rapido cuando son operaciones con matrices y operaciones muy extensas que demandan una gran carga computacional, por ende es mejor cambiar la base de memoria. generalmente los datos se guardan en sistemas de 32 bits, sin embargo existen mas sistemas de guardado, si tiene menos memoria tendrá una capacidad de guardar un numero mas pequeño, pero como a veces solo necesitamos números pequeños, enteros y positivos, entonces es mejor saber estos "trucos"

uint8()= es un entero de 8 bits que no tiene signo float()= flotante de 32 bits. también se puede denotar como float32() double()= flotante de 64 bits. también se puede denotar como float64() char()= un dato de letras de 32 bits string()= dato heredado de un char, union de varios char por tanto, gasta 32 bits por letra y por esto se le llama la unión de varios char

sabiendo esto, podemos ver que un dato tipo string es muy poco eficiente computacionalmente, ya que necesita 32 bits por cada letra que tenga lo cual gasta la memoria muy rápidamente