Universidad Nacional de Colombia
Facultad de Ciencias Exactas y Naturales
Sede Manizales

Relatoría Primer Corte

Informática III

Sofia Londoño Toro

16 de septiembre de 2022

Resumen

Mediante la presente relatoría se pretende mostrar los avances y aprendizajes del primer corte de la asignatura Informática III, los cuales se centrarán principalmente en la introducción a Phython, con temas como su entorno, su manejo de datos, sus funciones, entre otros. Esto con el fin de comprender mejor el funcionamiento de este lenguaje de programación y construir una base sólida en su manejo.

Clase 1: 24/08/2022

Instalación de los programas que se usarán en la materia: Visual Studio Code, git, creación de cuenta de GitHub. Se creó el repositorio en la nube: - Clases - Ejercicios - Informes

- CLASES: el cual contendrá el contenido de la clases impartidas durante todo el curso, estará compuesto por archivos con texto (como este) y prácticas de código.
- EJERCICIOS: Contendrá los ejercicios propuestos para cada clase, ejemplos y trabajo extra clase.
- INFORMES: Contendrá las relatorías y ejercicios de carácter entregable.

Se inició el repositorio desde Visual Studio, desde la función Source Control (icono 3 nodos), la U que aparece al lado de los archivos signifca que hay que actualizar (update) los archivos, esto significa que se almacenó en el 'Working Directory' para pasar al 'Stage Area' se le debe dar + y que aparezca A, para subirlo a 'Commited Area', se le debe dar commit, esto crea como un historial de hasta donde va el archivo, guarda lo que se hizo hasta el momento, de ahí para publicarlo se debe de dar click en el icono de nube.

Para que el computador reconozca el usuario de gitHub se debe abrir git Bash y escribir: git config –global user.name "nombre"
git config –globla user.email .^{em}ail"

Se empleará la función más simple de Python "print"que se usa para mostrar lo que está entre paréntesis en el terminal.

```
print("hola mundo") Cadena de texto
print(1) Entero
print(True) booleano
print(12.478) flotantes
print([1,2,3]) lista
print((1,2,3)) tupla
print('variable = ', 12, 'Este numero es importante')
```

Clase 2: 26/08/22

TIPOS DE DATOS EN PYTHON

BÁSICOS

BOOLEANOS: True, False. Trabaja con operaciones de compuertas lógicas
STRING: , " ",'casa', '123', ' " ' Para definir textos largos de varias lineas:

”’__-
__-”’

Cuando se suman str, lo que hace es concatenarlos (unir los strings)
ENTERO: -1000, 1000, 0
FLOTANTE: 12.5 , -100.0, 5.

- Al castear o convertir variables, cuando se pasa de flotante a entero, simplemente le quita el decimal, sin aproximar
- No se puede castear un string a número: entero o flotante, si el string contiene texto

COMPUESTOS

Sirven para combinar datos básicos, son contenedores de datos básicos.

LISTAS: [], [1,2,3] , [”, ”, ”], [’a’, 2, ’tres’, 4] ==>Se pueden mutar: cambiar el contenido
TUPLAS: (), (1,2,3), (”, ”, ”), (’a’, 2, ’tres’, 4) ==>Inmutables

Permiten operaciones de conjuntos matemáticos: Unión, Intersección ...

DICCIONARIOS:’CRISTIAN’ : 5, ’DANIELA’ : 0, ’SEBASTIAN’: 3 ==>Permite relacionar y enlazar variables

OPERACIONES

BOOLEANOS: +, AND, OR, NOR

ENTEROS Y FLOTANTES:

+,SUMA
-, RESTA
*, MULTIPLICACION
/, DIVISION
//, DIVISION ENTERA
**, POTENCIA
<, MENOR
>, MAYOR
==, COMPARAR
<=, MAYOR O IGUAL
>=, MENOR O IGUAL
!=, DIFERENTE

STRINGS:

+, CONCATENACION: Union de strings
string[index], indexado: saca el valor dado en ese indice

Algunos ejemplos:

`Declaramos los siguientes elementos y asignamos en variables De su elección:`

```
Enteros: 10, -100, 500, 200 ==>Luego restarlos de manera sucesiva

a= 10
b = -100
c = 500
d = 200
RESTA = a -( b - (c - d) )
print('resta de enteros: ', RESTA)

Flotantes: 100.0, 305.2, 400.3 ==>dividalos de manera sucesiva

f1, f2, f3 = 100.0, 305.2, 400.3
DIVISION = f1 / f2 / f3
print('División de flotantes', DIVISION)

Boleanos : True, False ==>primero sumelos, luego restelos

b1 = True
b2 = False
bsuma = b1 +b2 (se toma False como 0 y true como entero)
bresta =b1 -b2
print( 'Suma de booleanos: ', bsuma)
print( 'Suma de booleanos: ', bresta)

Strings: 'Cristina', 'Juanita' , , '', '"', '' sumelos, restelos ¿Que sucede?
s1 = 'Cristina'
s2 = 'Juanita'
s3 =
s4 = ''
s5 =  Esto no se puede hacer
s6 = '' ''
print('Se tienen los siguientes str: ', s1, s2, s3, s4, s6)
suma = s1 + s2 + s3
resta = s1 - s4 - s2 No se puede
print('suma de str >',suma)

Busque la manera de convertir (casteo):
Un entero a un flotante
Un flotante a un entero
Un flotante y un entero a un string
Un string a un entero y flotnte
Un booleano a un entero y flotante
Un entero y flotante

entero = 10
flotante = 11.5
string = '212'
booleano = True
print('Entero a Flotante = ', entero, float(entero))
print('Flotante a Entero = ', flotante, int(flotante)) No aproxima, solo quita el
```

```
decimal
print('Flotante a string = ', flotante, str(flotante))
print('Entero a string = ', entero, str(entero))
print('String a Flotante = ', string, float(string)) No deja convertir texto en
numero
print('String a entero = ', string, int(string)) No deja convertir texto en numero
print('Booleano a entero', booleano, int(booleano))
print('Booleano a flotante = ', booleano, float(booleano))
```

Clase 3: 31/08/22

TIPOS DE OPERADORES

Operador de asignación: =
Operadores aritméticos: +,-, *, /, //, **, %
Operadores lógicos: not, and, OR
Operadores de comparación: >, <, <=, >=, !=, ==
Operadores de pertenencia: in, not in
Operadores de conjuntos: | (unión), & (intersección), - (Resta)

Orden de operaciones: **, * /, +-

Código ejemplo

```
Operadores de asignación

a = 1
b = 2
c = 3

Operadores aritméticos

print("Suma =>", 1+2+3)
print("Multiplicación y suma =>", 1*2+3)
print("Suma y multiplicación =>", 1+2*3)
print("División entera =>", 5//3)
print("División entera =>", 0//3)
print('Residuo ', 5% 2)
print('Residuo ', 13% 9)
print("Potencia =>", 9**2)
print()''Raiz cuadrada'', 9**0.5)

print('concatenación =>', 'hola ' + 'mundo')
print('replicación =>', 'a' * 10 )
print('concatenación =>', [1] + [1,2,3])
print('repliación =>', [0] * 10 )

Operadores lógicos
```

```
print('and =>', True and False) Solo es verdadero si ambos son verdaderos
print('and =>', False and True)
print('or =>', True or False) Solo es falso si ambos son falsos
print('or =>', False or False)

¿Se pueden utilizar solo con booleanos?
print(1 and 1) Cualquier numero diferente de 0 es True y el 0 es False
print(1 and 0)
print(25 and 20) Devuelve el primero que lee de izquierda a derecha porque ambos
son True
print(0 and 100)
print(100 or 0)
print(100 or 0)
print(-2 or 20) Como ambos son verdaderos muestra el primero que lee el -2
print('Cristian' and 'Elias') Cualquier cada con al menos 1 caracter es verdadero
print('Unal' and ' ')

Para pensar:
'''
Desarrolle un algoritmo, que me imprima si un numero es mayor de 18, sin usar el
condicional if, utilice operadores lógicos para ello '''

OPERADORES DE COMPARACIÓN
print('mayor =>', 1 >2)
print('menor =>', 19 <0)
print('menor o igual =>', -1001 <= -1001)
print('igual o igual =>', 3 == -5)
print('mayor o igual =>', 19 >= 20)
print('diferente =>', 30 != 31)

print('Cristian' >'Elias') compara por orden alfabetico
print(True >False)
print([20,2] >[20,1]) Compara elemento a elemento

OPERADORES DE PERTENENCIA

print('a in holamundo =>', 'a' in 'holamundo')
print('A in holamundo =>', 'A' in 'holamundo')
print('hola in holamundo =>', 'hola' in 'holamundo')
print(' in hola mundo =>', ' ' in 'hola mundo')
print('1 in [1,2,3] =>', 1 in [1,2,3])
print('1 in ["1", "2", "3"] =>', 1 in ['1', '2', '3'])
print('3 not in 1234567890 =>', '3' not in '1234567890')
print('01 in 0 1 2 3 4 5 6 7 8 9 =>', '01' in '0 1 2 3 4 5 6 7 8 9 ')
```

Clase 4: 02/09/22

FUNCIONES INGEGRADAS

Entrada y salida: input(), print(), format()
Ayuda: help(), dir(), type()
Matematicas: abs(), round(), pow() .....
Conversiones: int(), float(), str(), complex()
bool(), set(), list(), tuple()

bin(), oct(), hex(), int()

Secuencias: range(), enumerate(), zip()

Operaciones con secuencias: len(), sum(), max(), min(), sorted(),
map(), filter()

Código Ejemplo

```
------ Funciones de entrada (input) y salida print -------

''' - Solicitar una clave y luego responder en pantalla si es correcta o no. clave
= ''Unal2022'' - Solicitar un numero y luego responder en pantalla si es mayor a 18
'''

clave-real = ''Unal2022''
clave-usuario = input(''Ingrese su clave: ")
print((clave-usuario == clave-real and "La clave es correcta") or
"La clave es incorrecta")

numero = 18
numero-usuario = int(input(''Ingrese un número : "))
print((numero-usuario >numero and "Mayor a 18") or "No es mayor a 18")

----- funciones de formateo de valores numericos =>format(valor, tipo)

formato cientifico
numero = 1028091280
formato-cientifico = format(numero, ''e'') Arroja el dato en notación científica
print(formato-cientifico)
formato-cientifico = format(numero, ".2e") Redondea el número a 2 decimales
de notación científica
print(formato-cientifico)

"
exprese el numero 0.000 000 029 213 en formato cientifico con 3 decimales
"
numero = 0.000000029213
print(''original = ", numero, çientifico =", format(numero, ".3e"))

formato flotante
numero = 12.1392
print(format(numero, ".2f")) Imprime el dato como flotante con 2 decimales
```

```
numero1 = 0.045
numero2 = 12.531

 formatear numero1 a flotante con un decimal
'''
print(format(numero1, ".1f")
---------Funciones de Ayuda (help, dir, type) -------
"
Determine el tipo de dato de los siguientes mostrados (type) "
dt1 = ''1'', ''2'' ==>'set'
dt2 = 1:2.items() ==>'dict-items'
dt3 = range(1,10) ==>'range'
dt4 = '1' ==>'str'
dt5 = 1 ==>'int'
dt6 = (1,) ==>'tuple'
print(type(dt1), type(dt2), type(dt3), type(dt4), type(dt5), type(dt6))
¿qué funcionalidades tienen los tipos de datos: int, float, str, set?
entero = 1
flotante = 2.54
cadena = "hola"
conjunto = 1,2,3,4

print(--Funcionalidades enteros ----")
for funcion in dir(entero):
print(funcion)

print(--Funcionalidades cadenas ----")
for funcion in dir(cadena):
print(funcion)



--------- funciones de conversion (bin, oct, hex, int)---------

"
Convertir a binario, octal y hexadecimal los siguientes numeros:
1, 8, 513
"


print("numero 1 a bin, oct, hex ==>", bin(1), oct(1), hex(1))
print("numero 8 a bin, oct, hex ==>", bin(8), oct(8), hex(8))
print("numero 513 a bin, oct, hex ==>", bin(513), oct(513), hex(513))

'''Convertir a decimal los numeros: 0b1000000001 0o1001 0x201'''

print(int("0b1000000001", 2))
print(int("0o1001", 8))
print(int("0x201", 16))
---- Funciones para secuencias -------
secuencia =>rango, lista, tupla .....
```

```
'''Crear una secuencia con los numeros del 1 al 10"
secuencia1 = [1,2,3,4,5,6,7,8,9,10]
secuencia2 = (i for i in [1,2,3,4,5,6,7,8,9,10])
secuencia3 = range(1, 11, 1) el primer valor si se toma, el segundo valor no se
toma
print(list(secuencia3))

"
Cree las siguientes secuencias
-->2,4,6,8,10,12 .... 50
-->0,3,6,9,12,15,18 .... 200
-->10, 9, 8 , 7, 0
-->numeros multiplos del 2 y 3 entre 100 y 3 ==>96,90,84....6
-->numeros multiplos de 15 entre 1000 al 900

"
print(list(range(2,51,2))) ==>Imprime una lista desde 2 hasta 50 de 2 en 2
print(list(range(0, 201, 3)) + [200]) ==>Imprime lista de 0 a 200 de 3
en 3 añadiendole el 200
print(list(range(10,-1,-1))) ==>Lista desde 10 hasta 0 de -1 en -1
print(list(range(96,5,-6)))
print(list(range(990,899,-15)))

"
Luego calcule el tamaño, maximo, minimo y la suma de los elementos de la última
secuencia creada
len(), min(), max(), sum()
"
lista = list(range(990,899,-15))
print(len(lista), min(lista), max(lista), sum(lista))

------ funciones de mapeo (map) y filtrado (filter)
map ==>Sirve para operar cada elemento de una lista
filter ==>Sirve para devolver unicamente ciertos datos de una lista
con una condición

secuencia = range(990,-1,-15)

"
mapear la anterior secuencia de la siguiente manera:
a) Multiplicar cada elemento por 3.1
b) Convertir los números a cadenas
c) Transformar los numeros al residuo entre 5 "
"
Filtrar la anterior secuencia de la siguiente manera:
a) Retener solo los números pares
b) Retener solo los numeros que al sumarles 5 sean pares
c) Retener solo los números cuyo primer digito sea par
"
```

```
solucion mapeado punto a
def multiplicacion(elemento): Puede devolver lo que interese hacer, no
necesariamente True o False
return elemento * 3.1
print(list(map(multiplicacion, secuencia))) ==>La función 'list' vuelve los datos
una lista

Punto B:

def Conver-Cadena(elemento):
return str(elemento)

print(list(map(Conver-Cadena, secuencia)))

Punto C:

def Res5(elemento):
return elemento% 5
print(list(map(Res5, secuencia)))

solucion filtrado punto a

def esPar(elemento): debe devolver True or False
return elemento% 2 == 0
print(list(filter(esPar, secuencia)))

Punto B:

def Sum5-Par(elemento):
return (elemento + 5)% 2 == 0

print(list(filter(Sum5-Par, secuencia)))

Punto C:

def PrimDigPar(elemento):
while elemento >9:
elemento = elemento // 10
return elemento% 2 == 0
print(list(filter(PrimDigPar, secuencia)))
```

Clase 5: 07/09/22

CONDICIONAL IF

Permite ejecutar un conjunto de sentencias, en caso de que una condición sea verdadera.

En caso de que la condición sea falsa, las sentencias contenidas son ignoradas

—-NOTACIÓN—

if <condición>:
<sentencia1>
<sentencia2>
<sentencia3>
.....

"Determine si un numero es mayor a 18, utilizando el condicional IF"

if <condicion>:
<sentencia1>
<sentencia2>
....
else:
<sentencias del else>
"Determine si un numero es mayor o no a 18, utilizando el condicional IF-ELSE"

if <condicion1>:
<sentencias1>
elif <condicion2>:
<sentencias2>
elif <condicion3>:
<sentencias3>
else:
<sentencias else>

Algunos ejemplos:

```
"Determine si un numero es mayor a 18,
utilizando el condicional IF
"
numero = 18
if numero >18: print("Numero es mayor a 18")

"
Determine si un numero es mayor o no a 18,
utilizando el condicional IF-ELSE
"
numero = 18
if numero >18:
print("Numero es mayor a 18")
else:
print("Numero no es mayor a 18")


"
Pedir a el usuario que ingrese 3 numeros,
luego, imprimir el numero mayor y el menor
"
```

```
numero1 = int(input(Ïngrese primer numero: "))
numero2 = int(input(Ïngrese segundo numero: "))
numero3 = int(input(Ïngrese tercer numero: "))

mayor = 0

if (numero1 >= numero2) and (numero1 >= numero3):
mayor = numero1
elif (numero2 >= numero1) and (numero2 >= numero3):
mayor = numero2
elif (numero3 >= numero1) and (numero3 >= numero2):
mayor = numero3

print(’’El numero mayor es ".format(mayor)) Format pone ’mayor’ en los corchetes
```

WHILE

Evalúa una condición, y en caso de ser verdadera ejecuta las sentencias contenidas en el ciclo.

Durante el ciclo la condición podría cambiar a falso, lo que ocacionaría que el ciclo termine en la siguiente ejecución.

——-NOTACION———

while <condicion>:
<sentencia1>
<sentencia2>
<sentencia3>

Código de ejemplos

```
1. Crear un ciclo infinito

while True:
print(’’Ciclo ejecutado")

’’ Crear un ciclo infinito, pero protegerlo, en caso de que se ejecute más de 100
veces"

contador = 0
while True:
print(’’ciclo ejecutado ". format(contador))
contador = contador + 1

if contador >100:
print(’’Contador superado Vamos a romper la ejecución")
break

’’Imprima los números del 20 al 50, utilizando el ciclo while"
```

```
contador2 = 20
while True:
print(contador2)
contador2 = contador2 + 1 =>contador2 += 1 (suma 1)
if contador2 >50:
break

Solución 2:
i = 20
while i <51:
print(i)
i +=1

"Mostrar en la pantalla los 10 primeros número de Fibonacci"
n = 2
fib1 = 1
fib2 = 1
print(fib1, fib2)
while n <10:
fibNext = fib1 + fib2
fib1, fib2 = fib2, fibNext
print(fibNext)
n += 1

’’Realice un programa que solicite una contraseña, si la contraseñaes incorrecta,
solicite la contraseña nuevamente. En caso de que si coincida. termine el ciclo
"

contraseña-usuario = str(input(’’Ingrese una contraseña: "))
contraseña-original = "1234"

while contraseña-usuario != contraseña-original:
print("Las contraseña no coinciden")
contraseña-usuario = str(input(’’Ingrese una contraseña: "))

print(’’contraseña correcta")
```