UNIVERSIDAD CATÓLICA DE TEMUCO
INGENIERÍA CIVIL EN INFORMÁTICA
FACULTAD DE INGENIERÍA

Programación I

Repaso contenido unidad 1

Profesor: Ignacio Lincolao Venegas

Ingeniería Civil Informática, UCT

Transformación de tipo de dato

Para convertir a cadena se utiliza la función str()

```
str(789) # “789”
str(True) # “True”
str(2+2) # “4”
```

Para convertir un entero o un string a punto flotante se utiliza la función float():

```
float(2) # 2.0
float(345) # 345.0
float(“159”) # 159.0
```

Transformación de tipo de dato

Para convertir un punto flotante, un string o boolean a número entero se utiliza la función int()

```
int(147.021) # 147
int(True) # 1
int(False) # 0
int(“16”) # 16
```

Para convertir un entero, decimal o un string, a booleano se utiliza la función bool(), es importante destacar que bool() devolverá falso cuando se ingresen valores vacíos o nulos, y para cualquier otro caso equivale a true:

```
bool(0) # False
bool(‘’)# False
bool(‘casa’) # True
bool(2) # True
```

Math

La librería math es una librería que ofrece funciones matemáticas principalmente para el manejo de los números reales. De el conjunto de funciones existe la función sqrt() el cual permite resolver la raíz cuadrada de un número.

```
import math
math.sqrt(16) # 4
math.sqrt(4) # 2
```

Otra función es la función math.pow(x,y) lo cual devuelve el resultado de X^Y:

```
import math
math.pow(2,3) # 8
math.pow(4,3) # 64
```

También posee variables constantes:

```
math.pi # 3.14192...
math.e # 2.71828...
```

Random

La biblioteca random contiene funciones que se relacionan con los valores aleatorios. Una de las funciones es el randint() el cual devuelve un número entero entre los valores indicados, incluyéndose.

```
import random
print(random.randint(1,5)) # puede devolver 1,2,3,4,5
```

La función **random()** genera un número decimal entre 0 y 1 (puede generar 0, pero no 1)

```
import random
print(random.random()) # 0.165145432
```

La función uniform() genera un número decimal entre a y b

```
import random
print(random.uniform(6, 9)) # 8.123154169841
```

fuente: https://www.mclibre.org/consultar/python/lecciones/python-biblioteca-random.html

Manejo simple de string

Longitud de un string

Para algunos programas se tendrá la necesidad de saber el número de caracteres que contiene un string, para esto utilizamos la función len() el cual devuelve el número de caracteres del string.

```
Palabra = len(“Hola mundo”) # 10
# Otra manera de usarlo es, es importante indicar que la salida de la función len es de
# tipo entero, por lo cual es necesario transformarlo a string para concatenarlo
print(“La palabra casa contiene ”+ str(len(“casa”))+ “ caracteres”)

a = “palabra”
b = “test”
len(a)>= len(b) # devuelve True devuelve True porque “palabra” contiene más caracteres que “test”
```

Manejo simple de string

Extracción un substring de un string

Se puede extraer una porción de un string a través del uso de posiciones del string, el primer carácter tiene la posición 0 y luego se va aumentando 1 en 1.

```
cadena = “Tareas diarias”
# T a r e a s   d i a r  i  a  s
# 0 1 2 3 4 5 6 7 8 9 10 11 12 13

print(cadena[7:]) # diarias
print(cadena[1:]) # areas diarias
print(cadena[:6]) # Tareas
print(cadena[7:10]) # dia
```

Manejo simple de string

Extracción un substring de un string

Extraer el primer y ultimo string

```
cadena = “Tareas diarias”
# T a r e a s   d i a r  i  a  s
# 0 1 2 3 4 5 6 7 8 9 10 11 12 13

print(cadena[0]) # T
print(cadena[-1]) # s
```

Operadores de asignación

Hasta ahora el operador de asignación que conocen es el =, pero existen también existen otros operadores lógicos.

=	Asigna el valor a la variable	`a = 10` `print(a) # 10`

+=	Suma un valor a la variable	`a = 10`
		`a+=5`
		`print(a) # 15`

-=	Resta un valor a la variable	`a = 10`
		`a-=5`
		`print(a) # 5`

***=**	Multiplica un valor a la variable	`a = 10`
		`a*=5`
		`print(a) # 50`

Operadores de asignación

Hasta ahora el operador de asignación que conocen es el =, pero existen también existen otros operadores lógicos.

/=	divide un valor a la variable	`a = 10`
		`a/=5`
		`print(a) # 2`

****=**	calcula el exponente del valor de la variable	`a = 10`
		`a**=5`
		`print(a) # 100000`

//=	calcula la división entera del valor de la variable	`a = 9`
		`a//=5`
		`print(a) # 1`

%=	devuelve el resto de la división del valor de la variable	`a = 9`
		`a%=5`
		`print(a) # 4`

Imprimir múltiples líneas con un solo print

```
texto = “””Esta es la primera línea
Esta es la segunda línea
Esta es la tercera línea“””
print(texto)
## Impre
# Esta es la primera línea
# Esta es la segunda línea
# Esta es la tercera línea
# ---------- Alternativa al uso de “”” para los saltos de línea
texto = “Esta es la primera línea\n Esta es la segunda línea\n Esta es la tercera línea”
print(texto)
# Esta es la primera línea
# Esta es la segunda línea
# Esta es la tercera línea

# El “\n” significa un salto de línea o nueva linea que equivale cuando escriben un documento y apretan enter
```

Errores comunes

Uno de los errores más comunes al utilizar print es concatenar una cadena con valores que no son string

```
numero = 16
print(“Tengo ”+ numero + “ Años.”)
## Imprimirá un error TypeError: can only concatenate str (not "int") to str
## Solución 1
## Se transforma a string el número para que pueda concatenar sin problema
print(“Tengo ”+ str(numero) + “ Años”)
## Solución 2
## la función print al separar con “,” se puede ingresar otros tipos de datos para que el los transforme
## a string y luego los concatena
print(“Tengo”,numero, “Años”)
```

Errores comunes

Otro error con el print es olvidar el uso de las comillas

```
numero = 16
print(“Tengo ”, numero , Años.”)
## Imprimirá un error SyntaxError: EOL while scanning string literal
## Solución 1
## Agregar las comillas
print(“Tengo ”, numero, “Años.”)
```

Errores comunes

Otro error común es utilizar una variable que no existe

```
numeros = 16
print(“Tengo ”, numero , “Años.”)
## Imprimirá un error NameError: name 'numero' is not defined
## Solución 1
## Escribir bien el nombre de la variable
print(“Tengo ”, numeros , “Años.”)
```

Errores comunes

Otro error común escribir mal nombre de las funciones

```
numeros = 16
prrint(“Tengo ”, numero , “Años.”)
## Imprimirá un error NameError: name 'prrint' is not defined
## Solución 1
## Escribir bien el nombre de la funcion
print(“Tengo ”, numeros , “Años.”)
```

Errores comunes

Otro error común que sucede con if, elif y else es olvidar utilizar “:”

```
numeros = 16
if numero>5
    print(“Numero es mayor a 5”)
## Imprimirá un error SyntaxError: invalid syntax
## Solución 1
## Agregar : al final de la declaración
numeros = 16
if numero>5:
    print(“Numero es mayor a 5”)
```

Errores comunes

Otro error común es el incorrecto uso de la sangría. Solo se debe aumentar la sangría cuando se utiliza “:”, luego debe volver a la sangría anterior

```
numeros = 16
if numero>5:
----if numero < 16:
--------mayor= numero
-------print(“Numero es mayor a 5”)
## Imprimirá un error IndentationError
## Solución 1
## ajustar correctamente la sangria
numeros = 16
if numero>5:
----if numero < 16:
--------mayor= numero
----print(“Numero es mayor a 5”)
## Solución 2
numeros = 16
if numero>5:
----if numero < 16:
--------menor= numero
--------print(“Numero es mayor a 5”)
```

Errores comunes

Otro error común es el mal uso de los paréntesis:

```
numeros = 16
if ((numero>5) and (numero < 16):
    print(“Numero es mayor a 5”)
## Imprimirá un error SyntaxError: invalid syntax
## Solución 1
## Revisar bien los inicios y cierres de los paréntesis
numeros = 16
if ((numero>5) and (numero < 16)):
    print(“Numero es mayor a 5”)
```

Errores comunes

Otro error común es al realizar las comparaciones y confundir número y textos iguales o al comparar palabras pensar que son iguales a pesar de tener letras mayúsculas y minúsculas diferentes.

```
numeros = 16
if numeros == ‘16’:
    print(“es igual a 16”)
else:
    print(“No es igual a 16”)
## Imprimirá un no es igual a 16
## Solución 1
## Ambos deben ser del mismo tipo de dato
numeros = str(16)
if numeros == ‘16’:
    print(“es igual a 16”)
else:
    print(“No es igual a 16”)
```

Errores comunes

Otro error común es al realizar las comparaciones y confundir número y textos iguales o al comparar palabras pensar que son iguales a pesar de tener letras mayúsculas y minúsculas diferentes.

```
numero = input(“Ingrese un numero”) # El usuario ingresa 16
if numero == 16:
    print(“es igual a 16”)
else:
    print(“No es igual a 16”)
## Imprimirá un no es igual a 16
## Solución 1
## Se recomienda que ambos deben ser del mismo tipo de dato
numeros = int(input(“Ingrese un numero”))
if numeros == 16:
    print(“es igual a 16”)
else:
    print(“No es igual a 16”)
```

Errores comunes

Otro error común es al realizar las comparaciones y confundir número y textos iguales o al comparar palabras pensar que son iguales a pesar de tener letras mayúsculas y minúsculas diferentes.

```
palabra= “Casa”
if palabra== ‘casa’:
    print(“Son iguales”)
else:
    print(“No son iguales”)
## Imprimirá No son iguales ya que poseen distintos caracteres, “C” es distinto a “c”
```

Uso del if y elif

```
numero = 16
if(numero >= 16):
    print(“El número es mayor o igual a 16”)
elif(numero < 32):
    print(“El número es menor a 32”)
else:
    print(“El número es mayor o igual a 32”)

## Mensajes
## “El número es mayor o igual a 16”

if(numero >=16):
    print(“El número es mayor o igual a 16”)
if(numero < 32):
    print(“El número es menor a 32”)
else:
    print(“El número es mayor o igual a 32”)


## Mensajes
## “El número es mayor o igual a 16”
## “El número es menor a 32”
```

Operador relacional != y <>

Ambos son operadores relacionales que permiten distinguir si los valores son distintos, pero él operador <> actualmente está obsoleto y fue removido en python 3, por lo que solamente funciona en python 2.

diferencias del & y del and (No entra en la prueba)

Diferencias entre el and y &, ambos sirve para realizar operaciones lógicas, pero el & es de un uso más avanzado.

and			Salida	&			Salida
True	and	True	True	True	&	True	True
True	and	False	False	True	&	False	False
7	and	9	9	7	&	9	1

diferencias del & y del and (No entra en la prueba)

El & al comparar dos números lo compara nivel de bits, en este caso el 7 a nivel de bits es -> 0000 0111 y el número 9 a nivel de bits es 0111 1001, el & los compara a través de bits por bits donde los 0 son False y los 1 True

0 0 0 0 0 1 1 1	7
0 1 1 1 1 0 0 1	9
0 0 0 0 0 0 0 1	1