**Lesson 2**

# -\*- coding: utf-8 -\*-

"""

Created on Tue Oct  4 15:04:15 2022

@author: elettrav

"""

# VARIABLE

# A variable name can be anything: names, numbers, or underscore. It cannot begin with a number.

# Cannot contain ! or %. Anything is valid, unless it is a Python keyboard: for, if, while, import are not ok.

name = "Beau"

\_name = "Elettra"

HEIGHT = 165

# 1ciao NO

# 123   NO

# test! NO

# test% NO

# EXPRESSIONS

# An expression is any code that returns value: ex. 1+2, "Beau"

# STATEMENTS

# Any operation on a value: ex. name = "Beau”; print(name)

# A program is a list of statements. They can be on the same line with ;

# COMMENTS

# Ignored line. Useful to add informations.

# INDENTATION

# Be careful with indentation. Everything indented belongs to a block.

# DATA TYPES

# Strings: ex. "Beau"

# It is possible to check the type

print(f"The type of name is {type(name)}")  # <class 'str'>

print(type(name) == str)  # True

print(isinstance(name, str))  #True

# Integers numers <class 'int'>

# Floating point <class 'float'>

# 2 in int, but I can make it a float:

number = 2

age = float(2)

print(f"The type of age is {type(age)}")

#I cannot convert string in numbers and viceversa

# OPERATORS

# = is the assignment operator

# Arithmetics operators are + - \* / % \*\* //

# \*\* to the power of

# // rounds up to the minor integer

age = 8

age += 8 #age = age + 8

print(f"Age is {age}")  # 16

# COMPARISONS

# a == b

# a != b

# a > b

# a <= b    --> True/False (Boolean data type)

# BOOLEAN OPERATORS

condition1 = True

condition2 = False

print(not condition1) #False

print(condition1 and condition2) #False

print(condition1 or condition2) #True

# OR returns the first not False value or the last one

print(0 or 1) #1

print("hi" or "hey") #hi

print([] or condition2) #False

# AND only evaluates the second argument if the first one is True

print(0 and 1) #0

print(1 and 0) #0

print(condition2 and "hey") #False

print("hi" and "hey") #hey

print([] and False) #[]

#BITWISE operators:

# is

# input

# TERNARY OPERATORS

def is\_adult(age):

  if age>18:

    return []

  else:

    return "not"

result = is\_adult(age)

print(f"The person is {result} an adult")

def is\_adult2(age):

  return [] if age>18 else "not" #This is a ternary operator. It is a shorter way with same result.

result2 = is\_adult2(age)

print(f"The person2 is {result} an adult")

# STRINGS

# It is possible to use both "" and ''

phrase = name + " is my name" #Remember to add space

print(phrase)

\_name += " is my name"

print(\_name)

# MULTILINES

print("""Sicily is

an island

in the center

of the Mediterranean

""") #Indentation will be reflected in the printed text.

# STRING METHODS

print(name.upper())      #BEAU

print(name.lower())      #beau

print("norway is the best country".title())      #Norway Is The Best Country

print("BeAu".islower())  #False

print(name.startswith("c"))  #to check if it starts with a specific substrong   #False

print(len(name))

print( "au" in name)  #True

# ESCAPING CHARACTERS

# How to use " as text

phrase2 = "He said \"I am going to the gym\""

print(phrase2)

#To go to the next line

phrase3 = "He said: \n\"I am going to the gym\""

print(phrase3)

# name = "Be\au"  --> Beu

# name = "Be\\au" --> Beau

# STRING CHARACTERS AND SLICING

word = "abcdefgh"

print(word[1]) #b

print(word[-1]) #h

print(word[1:2]) #b  It stops to the letter befor 2nd place

print(word[1:3]) #bc

print(word[3:]) #defgh

# BOOLEAN

done = True

print(type(done))  #<class 'bool'>

if done:

  print("yes")

else:

 print("no")   #yes

# Numbers are always True, expect zero

# Strings (and dictionaries) are false only when empty ""

# print(type(done) == bool)  ---> True

if 0:

  print("yes")

else:

 print("no")  #no

# ANY

# Returns True if at least one is True

book1 = True

book2 = False

read\_any = any([book1, book2])

print(read\_any)  #True

# ALL

# Returns True if all the values are True

read\_all = all([book1, book2])

print(read\_all)  #False

# COMPLEX NUMBERS

# Different possibilities to write complex numbers:

num\_complex = 2+3j

num = complex(2,3)

#To extract real or imaginary part:

print(num\_complex.real, num\_complex.imag)

print(num.real, num.imag)

# ABSOLUTE

num1 = abs(-5.5)

print(num1)  #5.5

# ROUND

print(round(5.5))  #--> 6

print(round(5.49)) #--> 5

print(round(5.49,1))  #It rounds with one decimal number --> 5.5

# ENUM

#To create constants

from enum import Enum

class State(Enum):

  INACTIVE=0

  ACTIVE=1

print(State.ACTIVE)        #State.ACTIVE

print(State.ACTIVE.value)  #1

print(list(State))         #Print all the values

# CONTROL STATEMENTS

# A way to check if something is True or False

condit = True

if condit == True:

  print("The condition is true")

else:

  print("The condition is not true")

# LISTS

# Essential Python data structure

dogs = ["Roger", "Syd"]  #It can be mixed types in the same list

print("Roger" in dogs)  #True

print(dogs[0])          #Roger

dogs[1] = "Rock"             #It will rewrite the list item

# dogs[:3], dogs[2:] etc...

dogs.append("Quincy")        #It is added at the end

dogs.extend(["Judah", 5])    #To add more elements at the end

# dogs += ["Judah", 5]

dogs.remove("Rock")

print(dogs.pop())    #Return and remove the last element

dogs.insert(2, "Test")

dogs[1:1]=["Test1", "Test2"]  #Adds more object in the place 1

print(dogs)

# SORTING LISTS

# dogs.sort()  will sort alfabethically and put upper case words first. In this case it is modify directly the list.

# dogs.sort(key=str.lower)  ignores upper case

dogscopy=dogs[:]  #to copy a list

print(sorted(dogs, key=str.lower))  #does not modify the original list

The type of name is <class 'str'>

True

True

The type of age is <class 'float'>

Age is 16

False

False

True

1

hi

False

0

0

False

hey

[]

The person is not an adult

The person2 is not an adult

Beau is my name

Elettra is my name

Sicily is

an island

in the center

of the Mediterranean

BEAU

beau

Norway Is The Best Country

False

False

4

True

He said "I am going to the gym"

He said:

"I am going to the gym"

b

h

b

bc

defgh

<class 'bool'>

yes

no

True

False

2.0 3.0

2.0 3.0

5.5

6

5

5.5

State.ACTIVE

1

[<State.INACTIVE: 0>, <State.ACTIVE: 1>]

The condition is true

True

Roger

5

['Roger', 'Test1', 'Test2', 'Quincy', 'Test', 'Judah']

['Judah', 'Quincy', 'Roger', 'Test', 'Test1', 'Test2']