PYTHON FOR DATA SCIENCE BASICS

Ana-María Dobre

based on EDX Course

SEP 2023

1. TYPES

Examples of different types (1)

Typecasting (changing types)
Example: convert int into a float: (2)
or convert float into int: (3)
or convert string into int: (4)

Convert string that contains a non-integer value into int - get an error: (5)

Convert int or float to string: (6)

Using a type command on a boolean value, we obtain the term bool: (7)

Cast a Boolean False to an integer or float, you will get a 0: (8)

Check URL (9) for other types in Python

(1) integers: 15
// finite, but large range of integers exists
real numbers: 15.9
// both negative & positive
strings: "Geronimo"
booleans: True and False

(2) float(2): 2.0

(3) int(1.1): 1
// be careful, some info is lost

(4) int('2'): 2

(5) int('A'): error

(6) str(1): "1"
str(1.1): '1.1'

(7) type(True): bool

(8) int(False): 0
int(True): 1

bool(0): False
bool(1): True

(9) https://www.python.org/

2. EXPRESSIONS & VARIABLES

Expressions: mathematical operations, like (10)
10, 20, 30, 40 are OPERANDS
Math symbols, in this case addition (+) are OPERATORS

Other examples of expressions: (11)

May use the double slash for integer division, where the result is rounded: (12)

Expressions in parenthesis are performed first: (13)

Variables:
- it's common to use _ to represent the start of a new word: (14)
- use meaningful variable names

(10) 10+20+30+40

(11) 25/5 => 5.0
25/6 => 4.166
// In both examples, both will result in a float

(12) 25//6 => 4

(13) (4+5)*6

(14) total_min

3. STRINGS (I)

A string is a sequence of characters.
A string is contained within single or double quotes.
We can bind or assign a string to another variable.

It's helpful to think of a string as an ordered sequence. Each element in the string can be accessed using an index. (15)

We can also use negative indexing with strings. The last element is given by the index -1: (16)

We can input a STRIDE value: (17)
// The 2 indicates we select every second variable

SLICING: return every second value up to index 4: (18)

(15) name="GERONIMO"
=> 01234567
name[0]: "G"
name[1]: "E"

(16) name[-1]: "O"
name[-2]: "M"
name[-3]: "I"

(17) name[::2] => "GRNM"

(18) name[0:5:2] => "GRN"

3. STRINGS (II)

Obtain string length with the len command: (19)

Concatenate or combine strings with the addition symbols: (20)

Replicate string by multiplying it by the number of times you want to replicate it: (21)
Note: the result is a NEW string (strings are IMMUTABLE)

"\" (backslashes) represent the beginning of escape sequences.
Escape sequences represent strings that may be difficult to input: (22)

Note: We can also place an "r" in front of the string

Examples of string methods: (23)

(19) len("Geronimo") => 8
// there are 15 elements

(20) "I call it" + " Geronimo"

(21) 3*"Geronimo "
=> "Geronimo Geronimo Geronimo "

(22) "I call it \n Geronimo"
=> I call it
Geronimo
"I call it \t Geronimo"
=> I call it Geronimo
"I call it \\ Geronimo"
=> I call it \ Geronimo
print(r"I call it \nGeronimo")

(23) A = "Geronimo"
B = A.upper() => "GERONIMO"
c = A.replace("ro", "RO") => "GeROnimo"
A.find("ni"): 5
// Note: the output is the first index of the sequence you would like to FIND
A.find("shaman drums"): -1
// if the substring is not found, the output is a negative one.