2

Data Science Practicum II 2021/22, Lesson 2

Marko Tkalčič

Univerza na Primorskem

Table of Contents

Variables

String

Strings Exercises

Data Collections

List

Exercises - List

Assignment

Variables

- variable names
  - has to be one word with no space
  - has to be generated with letters, numbers, and underscore
  - must not start with number

Variables

- variable names
  - has to be one word with no space
  - has to be generated with letters, numbers, and underscore
  - must not start with number

- types
  - Numbers
    - int (signed integers)
    - long (long integers, they can also be represented in octal and hexadecimal)
    - float (floating point real values)
    - complex (complex numbers)
  - String
  - List
  - Tuple
  - Dictionary

Naming Convention in Python

- use lowercase names
- use underscore(s) to separate multi-words names
- do not use a single character (except for counter/iterator)
- do not use general names
- do not use long names
- good names:
  - `product_vector`, `term_definition`, `dataset_columns`
- bad names:
  - `list_of_unibz_studs_partici_program_data_analytics`
  - `a`, `g`, `z`, `o`

type() function

- returns the type of a variable

```
my_num=33
type(my_num)
```

```
my_str="Hello World!"
type(my_str)
```

Table of Contents

Variables

String

Strings Exercises

Data Collections

List

Exercises - List

Assignment

Strings

- Create a string variable

```
my_str = "Hello world!"
```

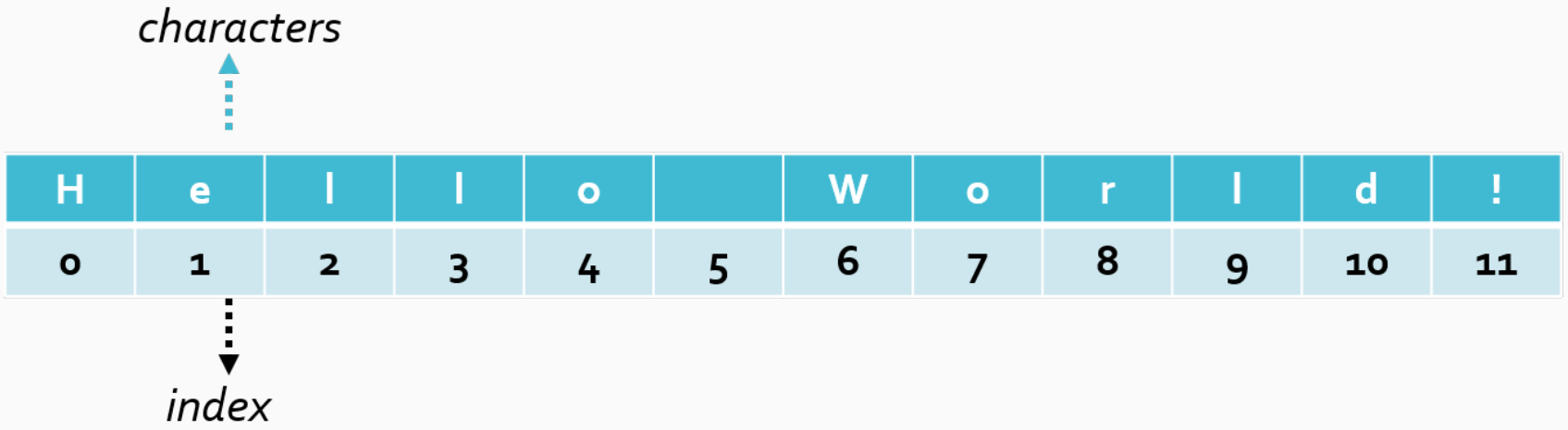
- what happened?

Strings

- Create a string variable

```
my_str = "Hello world!"
```

- what happened?

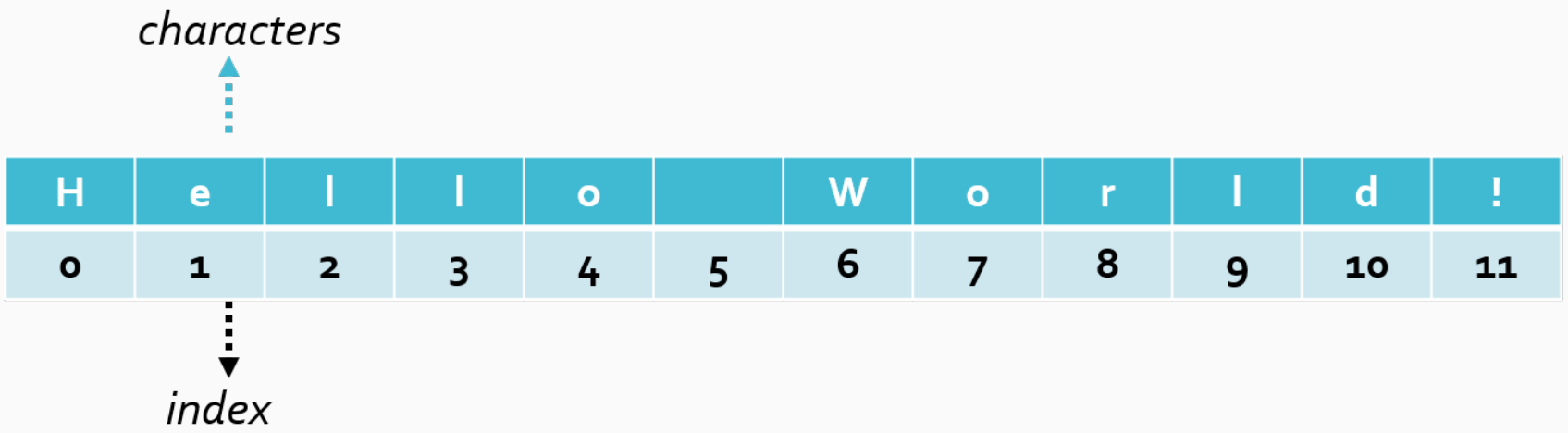


Strings

- Create a string variable

```
my_str = "Hello world!"
```

- what happened?



- you can access individual characters

```
my_str[0]
my_str[11]
```

```
my_str[0:4]
my_str[6:11]
```

Strings

- length

```
len(my_str) -> 12
```

Strings

- length

```
len(my_str) -> 12
```

```
my_str.upper()
```

Strings

- length

```
len(my_str) -> 12
```

```
my_str.upper()
```

```
my_str.find("world")
```

Strings

- length

```
len(my_str) -> 12
```

```
my_str.upper()
```

```
my_str.find("world")
```

```
new_str = my_str.replace("Hello!","Ciao!")
```

Table of Contents

Variables

String

Strings Exercises

Data Collections

List

Exercises - List

Assignment

Excercise 1 - String elements

- create a new notebook
- create a string variable `str` with the content `2018-12-05`
- print the length of the string
- print the 1st, 6th and 8th character

Excercise 1 - String elements

- create a new notebook
- create a string variable `str` with the content `2018-12-05`
- print the length of the string
- print the 1st, 6th and 8th character

```
In [13]: str1 = "2018-12-05"

In [11]: print(len(str1))
         10

In [10]: print(str1[0])
         print(str1[5])
         print(str1[7])
         2
         1
         -
```

Exercise - Manipulate Strings

- create a new string `str1 = "2018-05-12"`
- print the 5th character

Exercise - Manipulate Strings

- create a new string `str1 = "2018-05-12"`
- print the 5th character

```
print(str1[4])
```

Exercise - Manipulate Strings

- create a new string `str1 = "2018-05-12"`
- print the 5th character

```
print(str1[4])
```

- change the character with index 4 to another character

Exercise - Manipulate Strings

- create a new string `str1 = "2018-05-12"`
- print the 5th character

```
print(str1[4])
```

- change the character with index 4 to another character

```
str[4] = "a"
```

Exercise 2

```
In [15]: print(str1[4])
          -

In [16]: str[4] = "/"
         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call la
         st)
         <ipython-input-16-41baa3dc3161> in <module>()
         ----> 1 str[4] = "/"

         TypeError: 'type' object does not support item assignment

In [ ]:
```

!Strings are **immutable**

Exercise - Manipulate Strings w Functions

```
str1 = "2018-05-12"
```

- use the `find()` function to find the index of the character -

Exercise - Manipulate Strings w Functions

```
str1 = "2018-05-12"
```

- use the `find()` function to find the index of the character -

```
In [17]: str1 = "2018-05-12"

In [18]: str1.find("-")
Out[18]: 4
```

- the function `find()` is a member function of the string object

Exercise

- find the index of the second occurrence of the character -

Exercise

- find the index of the second occurrence of the character -

```
In [33]: ind1 = str1.find("-")
         ind2 = str1.find("-",ind+1)
         print(ind2)
```

Exercise

- replace the characters - in the string `str1` with the character /
- `replace()`

Exercise

- replace the characters - in the string `str1` with the character /
- `replace()`

```
str1.replace("-","/")
```

- use `print()` to print the new value of `str1`

Exercise

- replace the characters - in the string `str1` with the character /
- `replace()`

```
str1.replace("-","/")
```

- use `print()` to print the new value of `str1`

```
In [31]: print(str1)
Out[31]: '2018-05-12'
```

- what happened?

Exercise

- replace the characters - in the string `str1` with the character /
- `replace()`

```
str1.replace("-","/")
```

- use `print()` to print the new value of `str1`

```
In [31]: print(str1)
Out[31]: '2018-05-12'
```

- what happened?

- assign the return value of `str1.replace("-","/")` to a new variable and print out both of them

Exercise

- replace the characters - in the string `str1` with the character /
- `replace()`

```
str1.replace("-","/")
```

- use `print()` to print the new value of `str1`

```
In [31]: print(str1)
Out[31]: '2018-05-12'
```

- what happened?

- assign the return value of `str1.replace("-","/")` to a new variable and print out both of them

```
In [35]: str2 = str1.replace("-","/")
         print(str1)
         print(str2)

         2018-05-12
         2018/05/12
```

Exercise - substrings

- given the string `str1 = "2018-05-12"` print out:
  - a string composed of the first 3 characters
  - a string composed of the last 2 characters
  - a string composed from the third to the fifth character

Exercise - substrings

- given the string `str1 = "2018-05-12"` print out:
  - a string composed of the first 3 characters
  - a string composed of the last 2 characters
  - a string composed from the third to the fifth character

```
In [43]: str1 = "2018-05-12"

In [60]: print(str1[0:3])
         201

In [63]: ln = len(str1)
         print(str1[ln-2:ln])
         12

In [64]: print(str1[2:5])
         18-
```

- `[a:b]` b should be the index after the last

Exercise

- the string `str1 = "2018-05-12"` has the form `YYYY-MM-DD`
- transform it to the format `DD.MM.YYYY`

Exercise

- the string `str1 = "2018-05-12"` has the form `YYYY-MM-DD`
- transform it to the format `DD.MM.YYYY`

In [66]:
```
str1 = "2018-05-12"
yy = str1[0:4]
mm = str1[5:7]
dd = str1[8:10]
str2 = dd + "." + mm + "." + yy
print(str2)
```

```
12.05.2018
```

Table of Contents

Variables

String

Strings Exercises

Data Collections

List

Exercises - List

Assignment

Overview

- List: ordered and mutable
- Tuple: ordered and immutable
- Set: unordered and immutable and unindexed
- Dictionary: unordered and mutable and indexed

Table of Contents

Variables

String

Strings Exercises

Data Collections

List

Exercises - List

Assignment

List

- Data collection that is ordered and mutable
- Can have duplicates
- Represented with brackets []

```
my_list = [1,9,99]
```

- appending/retrieving

```
my_list.append(2019)
print(my_list)
```

```
my_list1[3] --> 2019
```

List

- multiple retrieval

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[1:3])
```

- try it! which elements are retrieved?

List

- multiple retrieval

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[1:3])
```

- try it! which elements are retrieved?

- from begginning to 4:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[:4])
```

List

- multiple retrieval

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[1:3])
```

- try it! which elements are retrieved?

- from begginning to 4:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[:4])
```

- from ‘cherry’ to end:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:])
```

Negative indeces

- negative indexes if you want to start the search from the end of the list -1 is the last one

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[-4:-1])
```

Negative indeces

- negative indexes if you want to start the search from the end of the list -1 is the last one

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[-4:-1])
```

- change value

```
thislist[1] = "blackcurrant"
```

List

- often we loop through a list (or any other collection structure for that matter)

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x)
```

List

- often we loop through a list (or any other collection structure for that matter)

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x)
```

- check if item is present

```
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
  print("Yes, 'apple' is in the fruits list")
```

List

- often we loop through a list (or any other collection structure for that matter)

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x)
```

- check if item is present

```
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
  print("Yes, 'apple' is in the fruits list")
```

- length

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

List - adding elements

- appending

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

- inserting

```
```thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)
```
```

List - deleting

- remove item

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

List - deleting

- remove item

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

- remove index or last

```
thislist = ["apple", "banana", "cherry"]
thislist.pop()
print(thislist)
```

List - duplicating

- is this correct? What happens?

```
list2 = list1
```

List - duplicating

- is this correct? What happens?

```
list2 = list1
```

- list2 will only be a reference to list1, and changes made in list1 will automatically also be made in list2

List - duplicating

- is this correct? What happens?

```
list2 = list1
```

- list2 will only be a reference to list1, and changes made in list1 will automatically also be made in list2

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy()
print(mylist)
```

```
thislist = ["apple", "banana", "cherry"]
mylist = list(thislist)
print(mylist)
```

List - merging

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

list3 = list1 + list2
print(list3)
```

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

for x in list2:
  list1.append(x)

print(list1)
```

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

list1.extend(list2)
print(list1)
```

Table of Contents

Variables

String

Strings Exercises

Data Collections

List

Exercises - List

Assignment

Exercises - List

- ordered, mutable
- can have duplicates
- functions
  - append
  - insert
  - len
  - count
  - sort/sorted
  - reverse
  - copy

Exercise - creating a List

- create a list with 10 elements, of which
  - at least one is int
  - string
  - float
  - at least two are duplicates

Exercise - creating a List

- create a list with 10 elements, of which
  - at least one is int
  - string
  - float
  - at least two are duplicates

```
In [7]: ll = [1,2,3,1.0,2.0,3.0,"one", "two","three"]

In [8]: print(ll)
        [1, 2, 3, 1.0, 2.0, 3.0, 'one', 'two', 'three']
```

Exercise - accessing list elements

- print the value of the third element of the list
- print the value of the last element
- print the value of the second to last

Exercise - accessing list elements

- print the value of the third element of the list
- print the value of the last element
- print the value of the second to last

```
In [9]: l1 = [1,2,3,1.0,2.0,3.0,"one", "two","three"]
        print(l1[2])
```

```
3
```

```
In [14]: print(l1[len(l1)-1])
         print(l1[-1])
```

```
three
three
```

```
In [13]: print(l1[len(l1)-2])
         print(l1[-2])
```

```
two
three
```

Exercise - accessing list elements

- assign to a new list the fifth to eight elements of the first list

Exercise - accessing list elements

- assign to a new list the fifth to eight elements of the first list

```
In [16]: l2 = l1[4:8]
         print(l2)

         [2.0, 3.0, 'one', 'two']
```

```
In [ ]:
```

Exercise - accessing list elements

- assign to a new list the elements from the fifth to the last element of the first list

Exercise - accessing list elements

- assign to a new list the elements from the fifth to the last element of the first list

```
In [17]: l2 = l1[4:]
         print(l2)
```

```
[2.0, 3.0, 'one', 'two', 'three']
```

Exercise - looping

- loop through the list `colors` and print all elements

```
colors = ["red", "green", "blue", "purple"]
```

Exercise - looping

- loop through the list `colors` and print all elements

```
colors = ["red", "green", "blue", "purple"]
```

```
i = 0
while i < len(colors):
    print(colors[i])
    i += 1
```

Exercise - looping

- loop through the list `colors` and print all elements

```
colors = ["red", "green", "blue", "purple"]
```

```
i = 0
while i < len(colors):
    print(colors[i])
    i += 1
```

```
for i in range(len(colors)):
    print(colors[i])
```

Exercise - looping

- loop through the list `colors` and print all elements

```
colors = ["red", "green", "blue", "purple"]
```

```
i = 0
while i < len(colors):
    print(colors[i])
    i += 1
```

```
for i in range(len(colors)):
    print(colors[i])
```

```
for color in colors:
    print(color)
```

Exercise - add elements

```
colors = ["red", "green", "blue", "purple"]
```

- append the element "yellow"
- insert at the second position the element "orange"

Exercise - add elements

```
colors = ["red", "green", "blue", "purple"]
```

- append the element "yellow"
- insert at the second position the element "orange"

```
In [25]: colors = ["red", "green", "blue", "purple"]
         colors.append("yellow")
         print(colors)

         ['red', 'green', 'blue', 'purple', 'yellow']

In [26]: colors.insert(1,"orange")
         print(colors)

         ['red', 'orange', 'green', 'blue', 'purple', 'yellow']
```

Exercise - update

change the value of the fourth element to "violet"

Exercise - update

change the value of the fourth element to "violet"

```
colors[3] = "violet"
print(colors)
```

```
['red', 'orange', 'green', 'violet', 'purple', 'yellow']
```

Exercise - remove

```
colors = ["red", "green", "blue", "purple"]
```

- remove the second element using the del function

Exercise - remove

```
colors = ["red", "green", "blue", "purple"]
```

- remove the second element using the del function

```
In [31]: colors = ["red", "green", "blue", "purple"]
         print(colors)
         del colors[1]
         print(colors)
```

```
['red', 'green', 'blue', 'purple']
['red', 'blue', 'purple']
```

Exercise - Sort

```
l1 = [2,5,3,8,7,7,4,5,9,8]
```

- create a new list `l2` using the function `sorted()`
  - print l1 and l2
- create a new list `l3` using the function `sort()`
  - print l1 and l2
- what do you observe?

Exercise - Sort

```
l1 = [2,5,3,8,7,7,4,5,9,8]
```

- create a new list `l2` using the function `sorted()`
  - print l1 and l2
- create a new list `l3` using the function `sort()`
  - print l1 and l2
- what do you observe?

```
l1 = [2,5,3,8,7,7,4,5,9,8]
l2 = sorted(l1)
print(l1)
print(l2)
l3 = l1.sort()
print(l1)
print(l3)
```

```
[2, 5, 3, 8, 7, 7, 4, 5, 9, 8]
[2, 3, 4, 5, 5, 7, 7, 8, 8, 9]
[2, 3, 4, 5, 5, 7, 7, 8, 8, 9]
None
```

Exercise

- make a list `l1` that has eight color names
- from the list `l1` make the list `l2` with color names that are shorter or equal to 4 characters

Exercise

- make a list l1 that has eight color names
- from the list l1 make the list l2 with color names that are shorter or equal to 4 characters

```
l1 = ["red", "blue", "green", "yellow", "brown", "violet", "white", "black"]
l2 = []
for color in l1:
    if len(color)<5:
        l2.append(color)
print(l1)
print(l2)
```

```
['red', 'blue', 'green', 'yellow', 'brown', 'violet', 'white', 'black']
['red', 'blue']
```

Table of Contents

Variables

String

Strings Exercises

Data Collections

List

Exercises - List

Assignment

Assignment

- given the list

```
my_list = [2,5,7,3,8,9,5,7,4,6,2,3,6,8,7,6,9,7,4,5,6,3]
```

- write the code that generates a list `my_ind`, which contains the indeces of the values from `my_list` that are bigger than 7 or smaller than 3

References

Part of the material has been taken from the following sources. The usage of the referenced copyrighted work is in line with fair use since it is for nonprofit educational purposes.

- https://stackoverflow.com/questions/6200910/relationship-between-scipy-and-numpy