Cmpe 150 Lab 8: Lists and Tuples

Up to Now

- We used a new variable for storing each piece of information, yet some are actually related. e.g., number of days in months

- Why don’t we store them together in an ordered manner?

Python Syntax for List and len

- empty_list = []
- non_empty_list = [4, True, 4, "A string", 643, 'Another String', some_variable]
- length = len(my_list)

Lists of Lists Also Possible

- A list can be an element of another list as well.

- nested_lists = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

Adding an Element to the List

- Use append function -> x.append(new_value)
- Before append, [3, 6, 7]
- After append(643) -> [3, 6, 7, 643]

Accessing the Elements of a List

- print(my_list[index]) or second_value_in_the_list = my_list[1]

- Again, be careful since counting starts from 0

Accessing the Elements of a List (cont.)

- Using negative indexes is also possible, just like the string case. In fact, we can think of any string as a list of characters.

- last_item = a_list[-1]

List Slicing

- Obtain some portion of our list by specifying the start and end (exclusive) indexes. x[start:end]

- All the tricks we had for strings are still valid, like the one below.

  first_five_elements = long_list[:5]

Changing Elements of a List

- Unlike strings, the elements of a list can be changed.
- z[3] = new_value
- z[2:4] = [9, 10]

+ and *

- We can concatenate two lists using the + operator

  [-1, -2, -3] + [-4, -5] -> [-1, -2, -3, -4, -5]

- We can obtain the repeated version of a list by using *

  [0, True] * 4 -> [0, True, 0, True, 0, True, 0, True]

remove and pop

- x.remove(val) removes the first occurrence of the val element in the list.
- x.pop(index) removes the element in the given index.

index and count

- first_occurence_index = my_list.index(search_item)
- n_occurences = my_list.count(search_item)

reverse and sort

- a_list.reverse()

- my_list.sort() Be careful; it does not return something; instead, it changes the ordering in the list.

List Comprehension and in

- A nice way to define a list, which would require a loop, within a single line of code

  squared_numbers_list = [i*i for i in range(1, 11)]

- el in my_list to check if an element is in the list.

Using a Loop with Lists

- for i in range(len(List)):

  print(List[i])

- for element in List:

  print(element)

Using a Loop with Lists (Cont.)

- Of course, it is also possible to use a while loop iterate over the list.

```
index = 0
while index < len(List):
    …
    index += 1
```

Using a list with sum(), max() and min()

- max_val_in_the_list = max(my_list)
- min_val_in_the_list = min(my_list)
- summation = sum(List) Try for ['Word1', 'Word2', 'Word3']

Converting a Different Type to List

- It is convenient for string cases.
- list_representation_of_str = list('A nice string With different characters.')

- ['A', ' ', 'n', 'i', 'c', 'e', ' ', 's', 't', 'r', 'i', 'n', 'g', ' ', 'W', 'i', 't', 'h', ' ', 'd', 'i', 'f', 'f', 'e', 'r', 'e', 'n', 't', ' ', 'c', 'h', 'a', 'r', 'a', 'c', 't', 'e', 'r', 's', '.']

Tuples

- example_tuple = (item1, item2, item3)

- They are very similar to lists but not mutable; in other words, we cannot add any new element or change the value of the existing one.

Tuples (Cont.)

- example_tuple = (item1, item2, item3)

- print(example_tuple[-1])

Tuples (Cont.)

- When we return multiple values from a function, it actually returns a tuple.

- Try to assign that to a single variable to see the type.

Sorted

- Sorting is an essential operation to solve various kinds of problems in computing. sorted is a built-in function in Python. Given a list, it does not change the ordering of the original list, yet it returns the sorted version.

- sorted(a_list, key=Function, reverse=Boolean)
  - key and reverse are optional

Sorted (Cont.)

- Elements of a list can be numbers, strings, or tuples as well.
- Example usage: Sorting the students according to their ages.

Bonus Content: Absolute Value

- abs_val = abs(my_variable)

- Example

  abs(5) -> 5

  abs(-3) -> 3

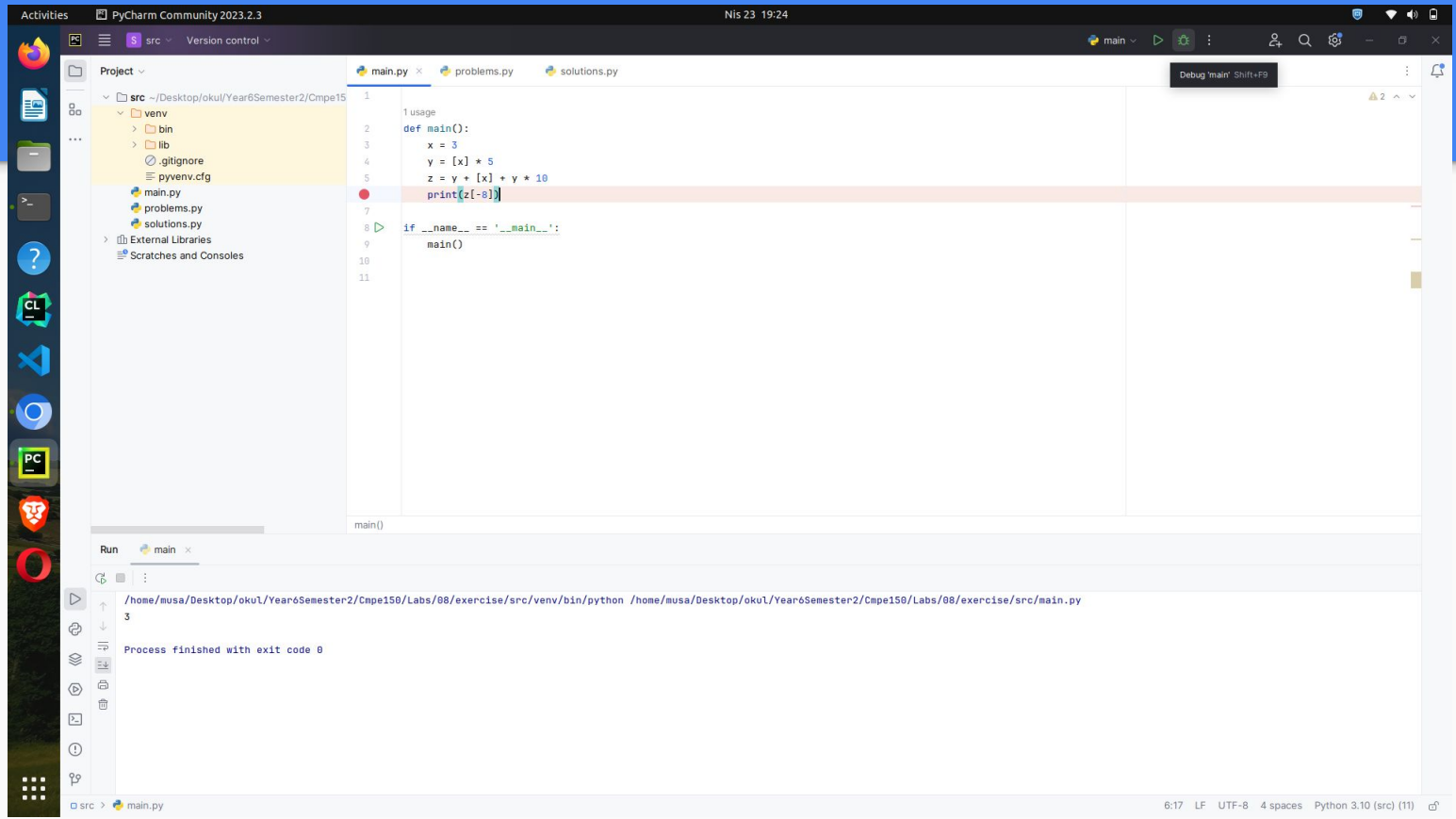
Tips About PyCharm

- Fast commenting: Ctrl + /

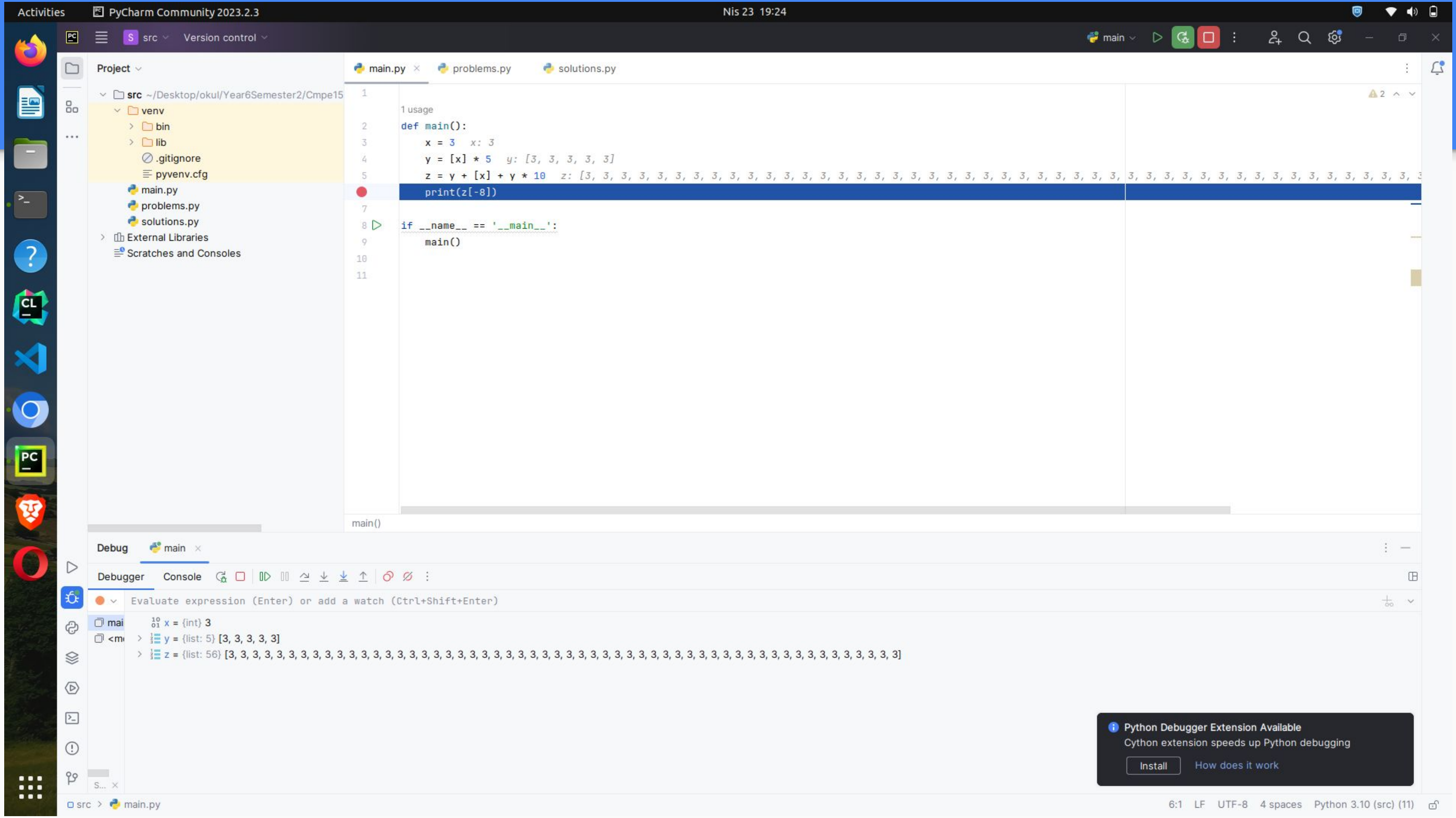
- You can use the PyCharm Python console or main.py file to conduct experiments.

PyCharm Python Console

Debugging


Activities
PyCharm Community 2023.2.3
Nis 23 19:24
src
Version control
main
Debug 'main' Shift+F9
Project
src ~/Desktop/okul/Year6Semester2/Cmpe15
venv
bin
lib
.gitignore
pyvenv.cfg
main.py
problems.py
solutions.py
External Libraries
Scratches and Consoles
main.py
problems.py
solutions.py
1 usage
def main():
x = 3
y = [x] * 5
z = y + [x] + y * 10
print(z[-8])
if __name__ == '__main__':
main()
main()
Run
main
/home/musa/Desktop/okul/Year6Semester2/Cmpe150/Labs/08/exercise/src/venv/bin/python /home/musa/Desktop/okul/Year6Semester2/Cmpe150/Labs/08/exercise/src/main.py
3
Process finished with exit code 0
src
main.py
6:17
LF
UTF-8
4 spaces
Python 3.10 (src) (11)

Debugging (Cont.)



Debugging (Cont.)

- It is also possible to use the console to run arbitrary code.
- In addition, F7 and F8 for different modes to proceed

Thanks

Any questions?

References

1. https://www.programiz.com/python-programming/list