Προγραμματισμός και Εφαρμογές με την Python Turtle

Δαδαλιάρης Αντώνιος (dadaliaris@uth.gr)

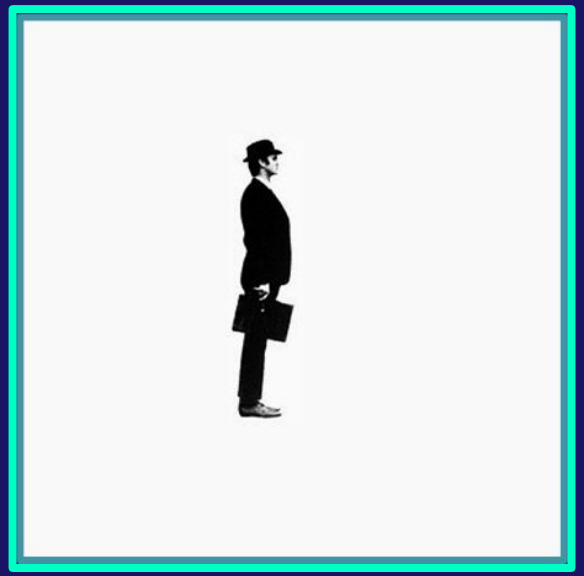
Περιεχόμενα

01

Python Basics Recap

“Pythonic” History

- Created by Guido Van Rossum in 1989 as a Christmas holiday project. . . .
- Current Version: 3.10.1
- Named after Monty Python



Python Characteristics

- Dynamic
- Open-source
- High-level
- Interpreted
- Object-oriented
- General-purpose
- OS agnostic
- Multi-Paradigm
- Frameworks
- Extendable

Python Characteristics

- Dynamic
  - Data can be modified dynamically during execution.
- Open-source
- High-level
- Interpreted
- Object-oriented
- General-purpose
- OS agnostic
- Multi-Paradigm
- Frameworks
- Extendable

Python Characteristics

- Dynamic
- Open-source
  - OSI licensed
- High-level
- Interpreted
- Object-oriented
- General-purpose
- OS agnostic
- Multi-Paradigm
- Frameworks
- Extendable

Python Characteristics

- Dynamic
- Open-source
- High-level
  - Syntactic and grammar rules that enhance readability & understanding.
- Interpreted
- Object-oriented
- General-purpose
- OS agnostic
- Multi-Paradigm
- Frameworks
- Extendable

Python Characteristics

- Dynamic
- Open-source
- High-level
- Interpreted
  - “Serial” command execution.
- Object-oriented
- General-purpose
- OS agnostic
- Multi-Paradigm
- Frameworks
- Extendable

Python Characteristics

- Dynamic
- Open-source
- High-level
- Interpreted
- Object-oriented
  - Classes
  - Objects
  - “Loose” encapsulation rules
- General-purpose
- OS agnostic
- Multi-Paradigm
- Frameworks
- Extendable

Python Characteristics

- Dynamic
- Open-source
- High-level
- Interpreted
- Object-oriented
- General-purpose
  - Various applications
- OS agnostic
- Multi-Paradigm
- Frameworks
- Extendable

Python Characteristics

- Dynamic
- Open-source
- High-level
- Interpreted
- Object-oriented
- General-purpose
- OS agnostic
  - Available for almost all operating systems.
- Multi-Paradigm
- Frameworks
- Extendable

Python Characteristics

- Dynamic
- Open-source
- High-level
- Interpreted
- Object-oriented
- General-purpose
- OS agnostic
- Multi-Paradigm
  - Procedural programming
  - Imperative programming
  - Object-oriented programming
  - Functional programming
- Frameworks
- Extendable

Python Characteristics

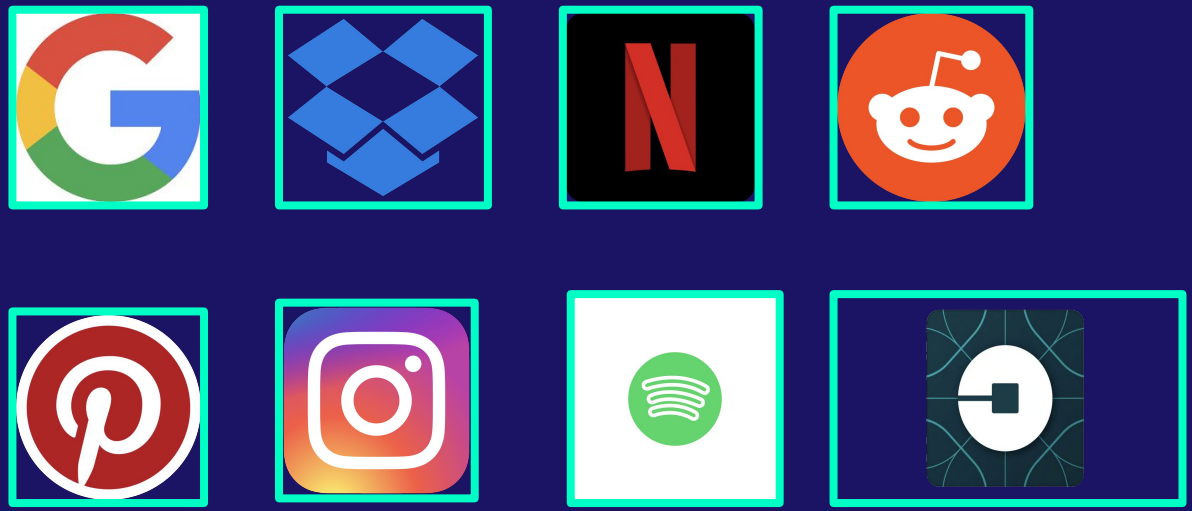
- Dynamic
- Open-source
- High-level
- Interpreted
- Object-oriented
- General-purpose
- OS agnostic
- Multi-Paradigm
- Frameworks
  - Flask
  - Django
  - . . . . .
- Extendable

Python Characteristics

- Dynamic
- Open-source
- High-level
- Interpreted
- Object-oriented
- General-purpose
- OS agnostic
- Multi-Paradigm
- Frameworks
- Extendable
  - Cython
  - Jython
  - Iron Python
  - . . . . .

Applications built with Python

- Google
- Instagram
- Dropbox
- Pinterest
- Spotify
- Uber
- Reddit
- Netflix



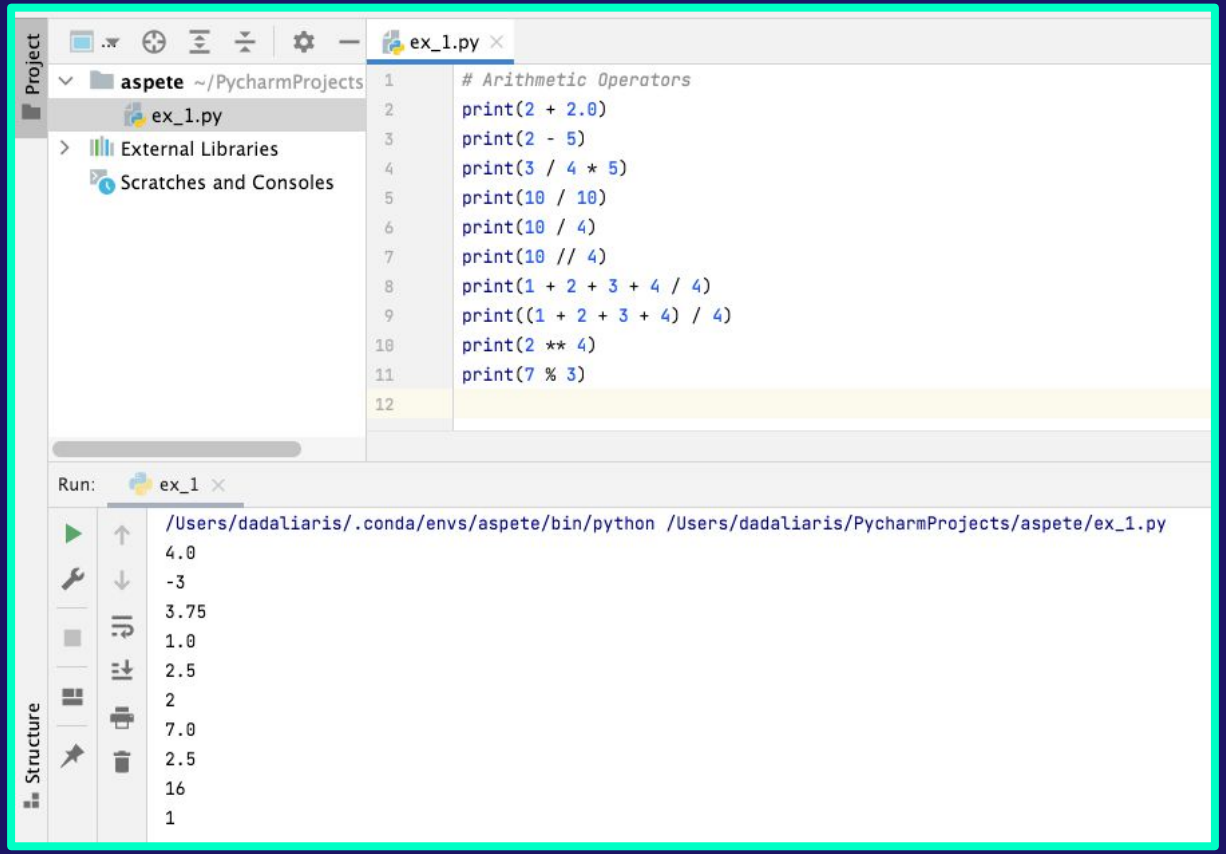
Variables & Numbers (1)

- **Python Console (IDLE, Spyder, ...):**
  - We can perform any arithmetic calculation directly at the console.
  - We can also execute various commands directly at the console.
  - The console is a powerful tool used mainly for quick references and serious debugging.

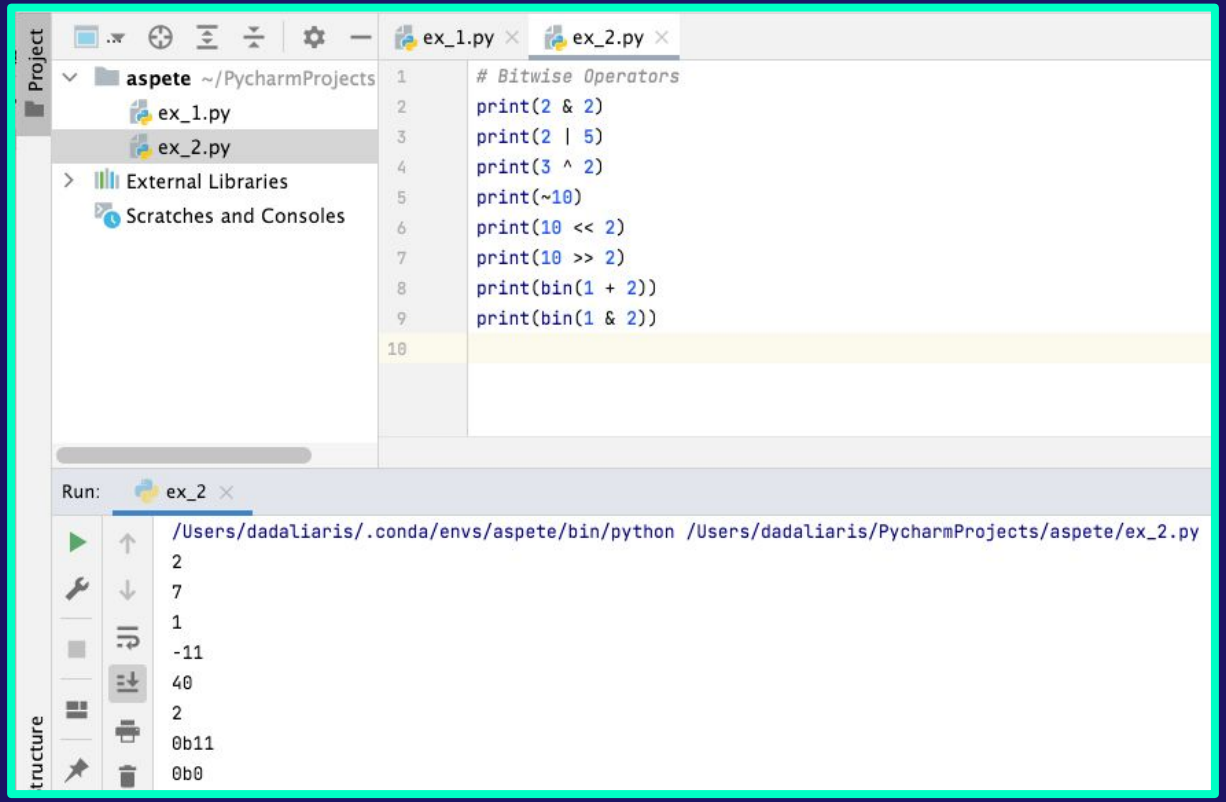
- **Operators:**
  - Arithmetic operators
  - Comparison operators
  - Logical operators
  - Bitwise operators
  - Assignment operators
  - Identity operators
  - Membership operators

Priority

Variables & Numbers (2)



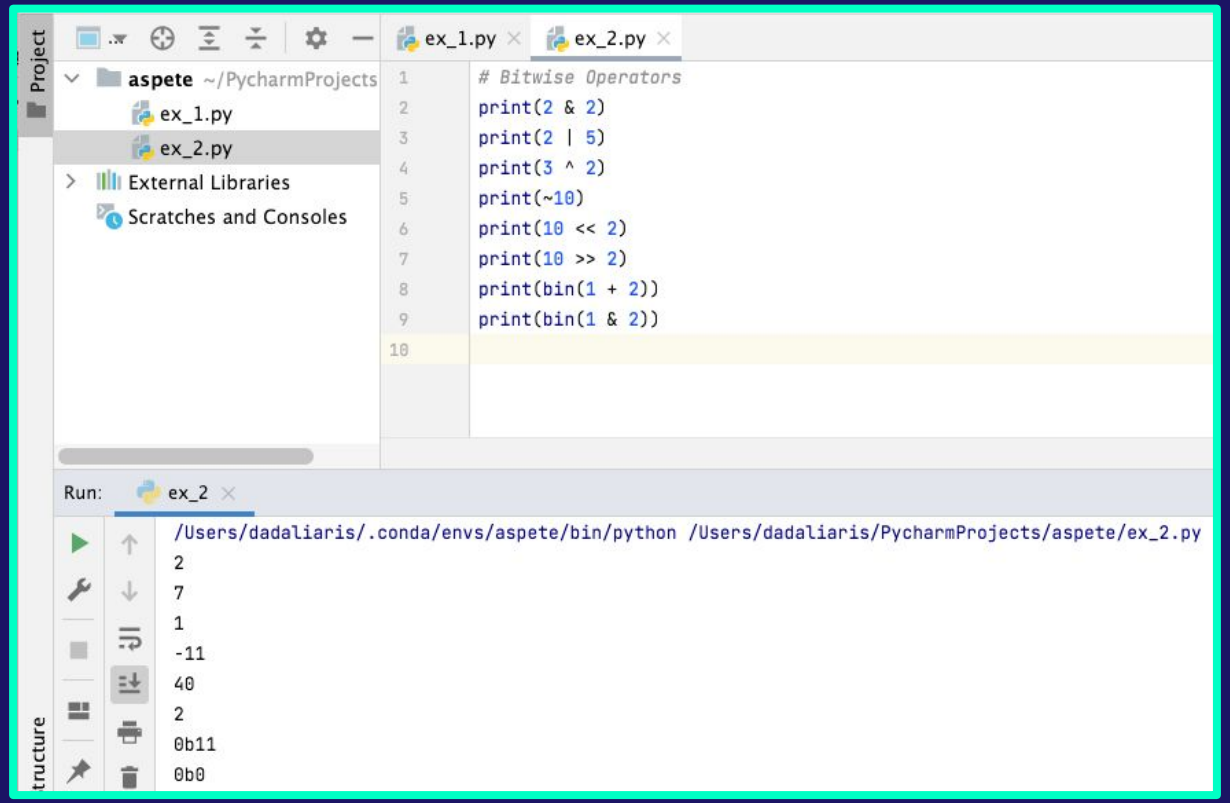
Variables & Numbers (3)



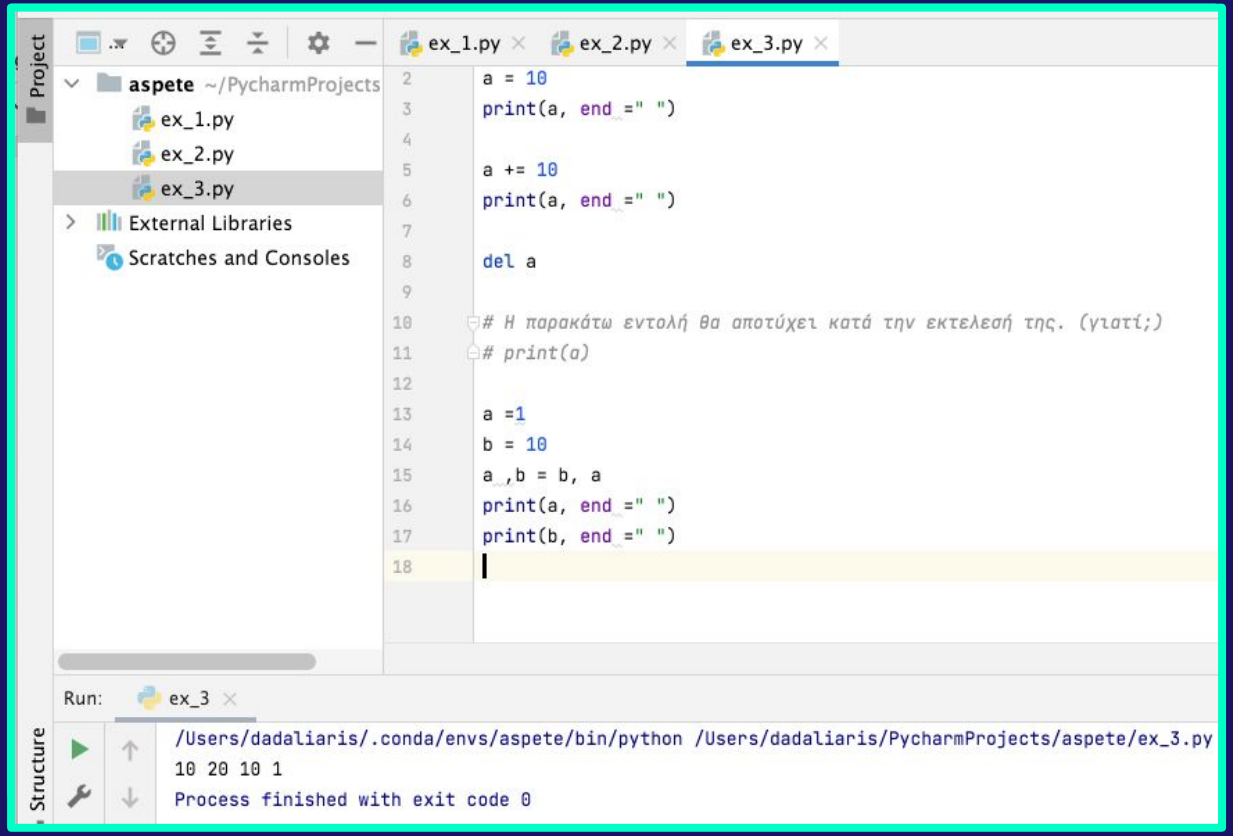
Variables & Numbers (4)

- **Variable:** a reserved memory location to store values.
- **Variable Properties:**
  - **Name**
    - Naming Rules: case-sensitive
      - names should always begin with a letter
      - letters, numbers, underscores
      - special symbols(*, -, /, #, ...)
      - reserved words:
        - False, None, True, and, as, class, continue, def, del, elif, finally, for, from, global, if, is, lambda, nonlocal, not, or, return, try, while, with, yield, import, else, assert, except, in, pass, break, raise
  - **Value:** various available data types
    - Integers: 1, 10, 232, 34534, ...
    - Floats: 3.14, 100.12345
    - Complex: 10 + 5j
    - Boolean: True, 1, False, 0
    - List: [0, 1.0, “two”], [1, 2, [3, 4, 5]]
    - Set: {1, 2, 3, “4”}
    - Tuple: (1, 2, “three”)
    - Dictionary: {1: “Adam”, 2: “Beatris”}

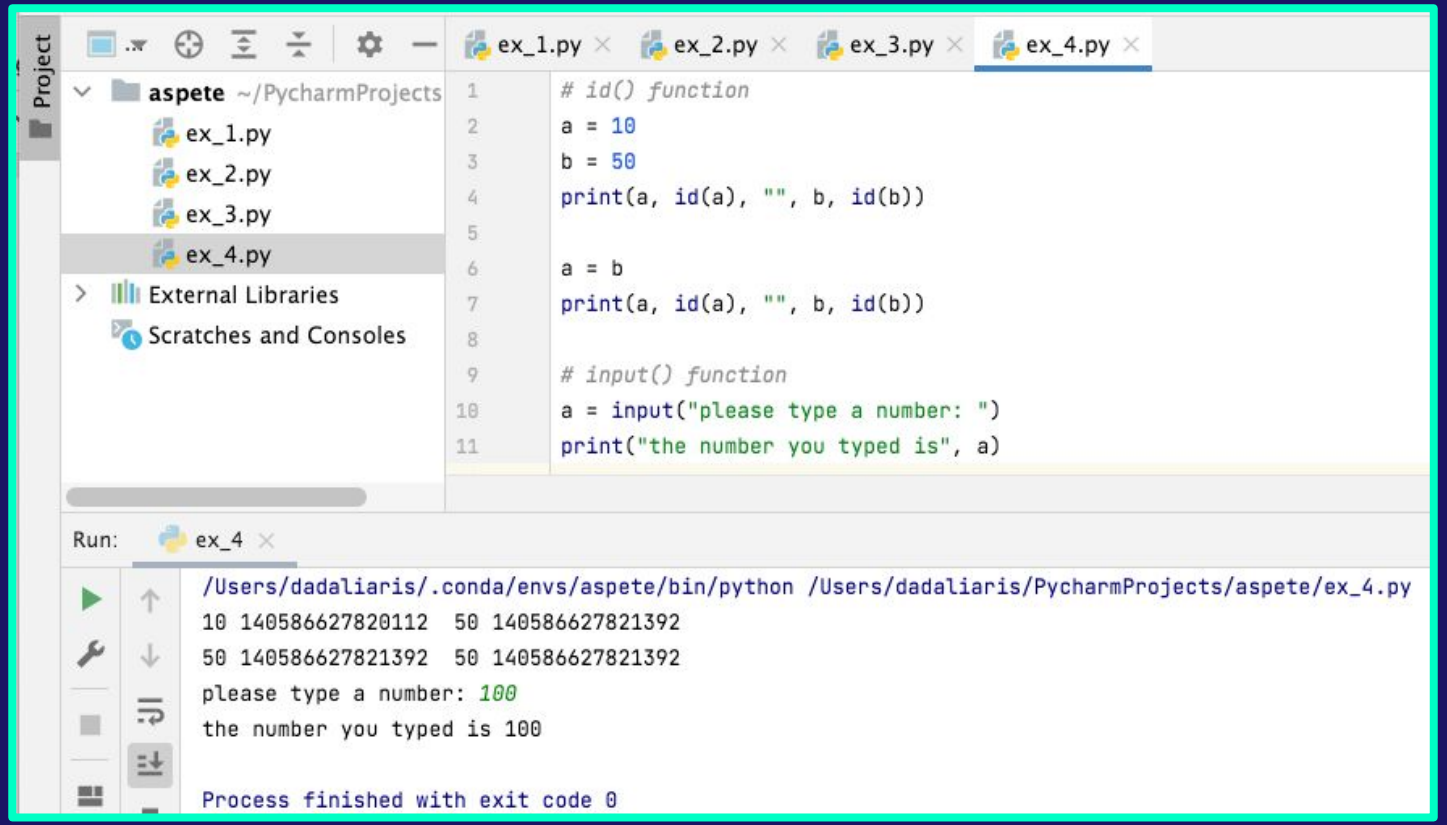
Variables & Numbers (5)



Variables & Numbers (6)



Variables & Numbers (7)

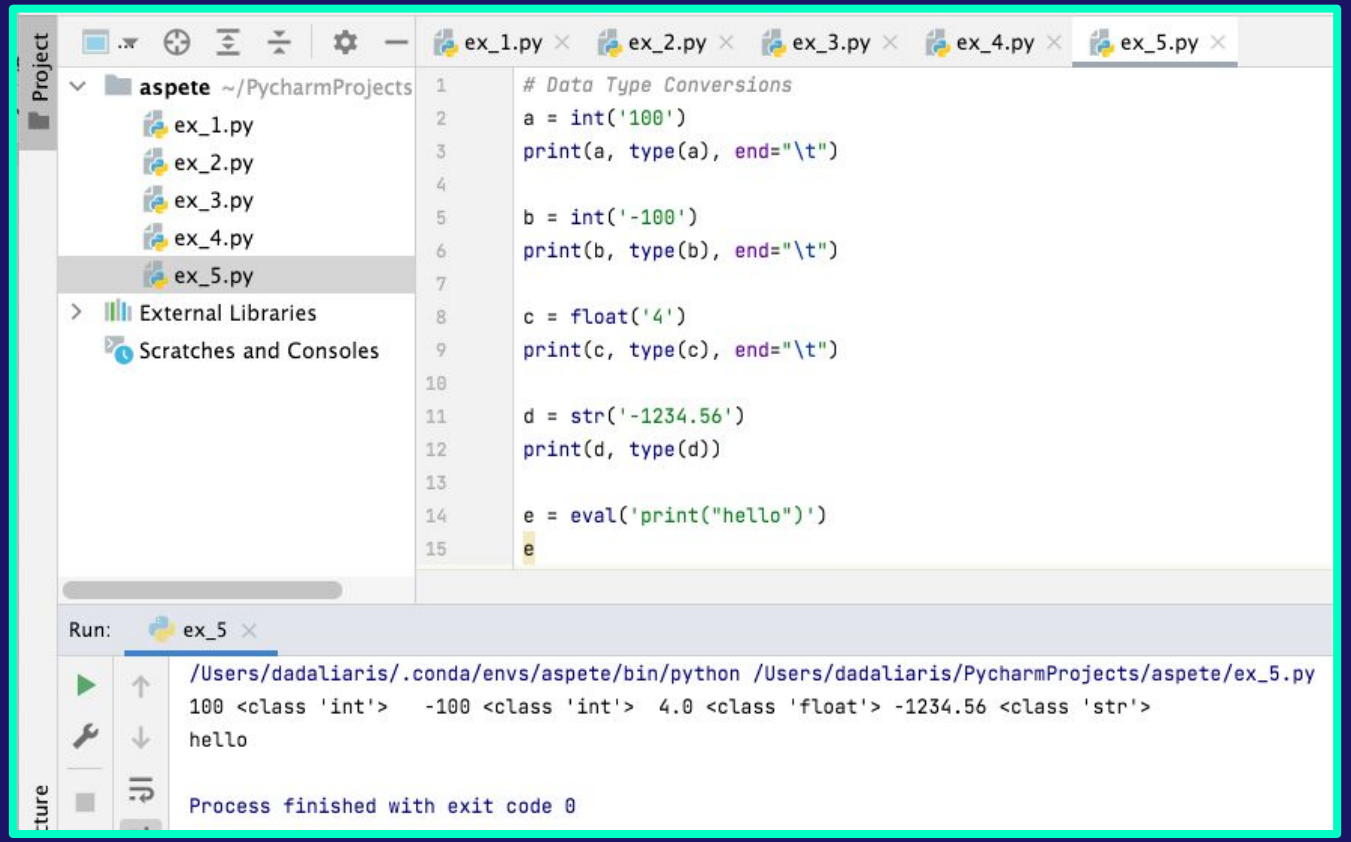


Variables & Numbers (8)

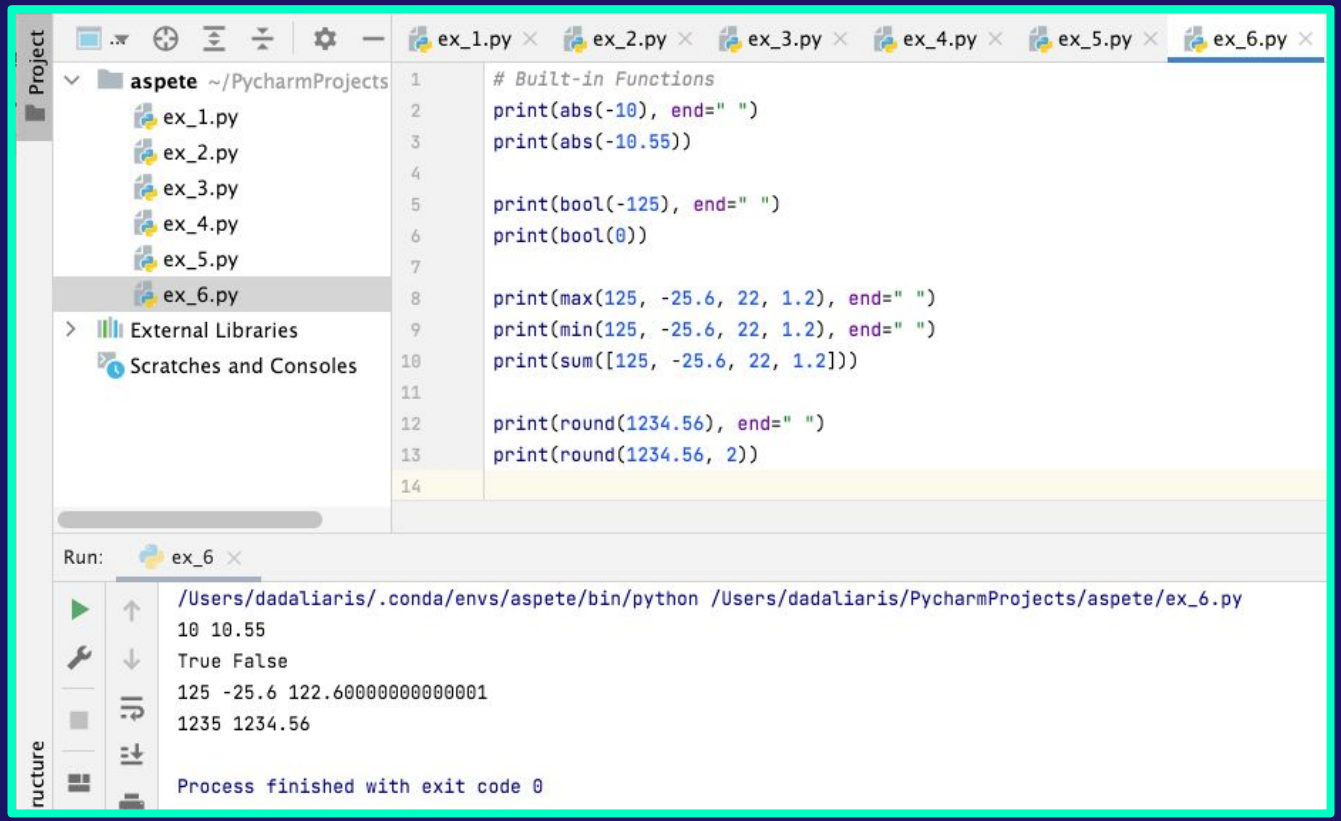
- **Data Type Conversions:**
  - int(a, base=n)
  - float()
  - complex(a b)
  - str()
  - eval(text)

- **dir(__builtins__):**
  - Errors
  - Warnings
  - Interrupts
  - Keywords ('True', 'False', ...)
  - Functions ('abs', 'all', 'any', 'eval', 'id', 'input', 'len', 'max', 'min', ...)

Variables & Numbers (9)



Variables & Numbers (10)



Variables & Numbers (11)

- Built-in Libraries:
  - math
    - acos, acosh, asin, asinh, atan, atan2, ceil, copysign, cos, cosh, degrees, e, erf, erfc, exp, expml, fabs, factorial, floor, fmod, frexp, fsum, gamma, gcd, hypot, inf, isclose, isfinite, isinf, isnan, ldexp, lgamma, log, log10, loglp, log2, modf, nan, pi, pow, radians, sin, sinh, sqrt, tan, tanh, tau, trunc
  - random
    - random.seed(number)
    - random.seed() (uses the system clock by default)
    - random.randint(from_a, to_b)
    - random.random()
  - . . . . .
- "External" Libraries - Modules:
  - Fractions
  - pandas
  - tkinter
  - . . . .

Flow Control / Conditional Statements (1)

- **It all comes down to booleans . . . .**
  - Values:
    - True (1)
    - False (0)
  - Boolean numbers can be used in classic arithmetic operations:
  - Example: a = True + 2-> a=3

==	1 == 2	False
!=	1 != 2	True
>	5 > 10	False
<	5 < 10	True
>=	4 >= 4	True
<=	4 <= 10	True

Flow Control / Conditional Statements (2)

- Logical Operators:
  - and
  - or
  - not

a	b	a and b	a or b	not(a)
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

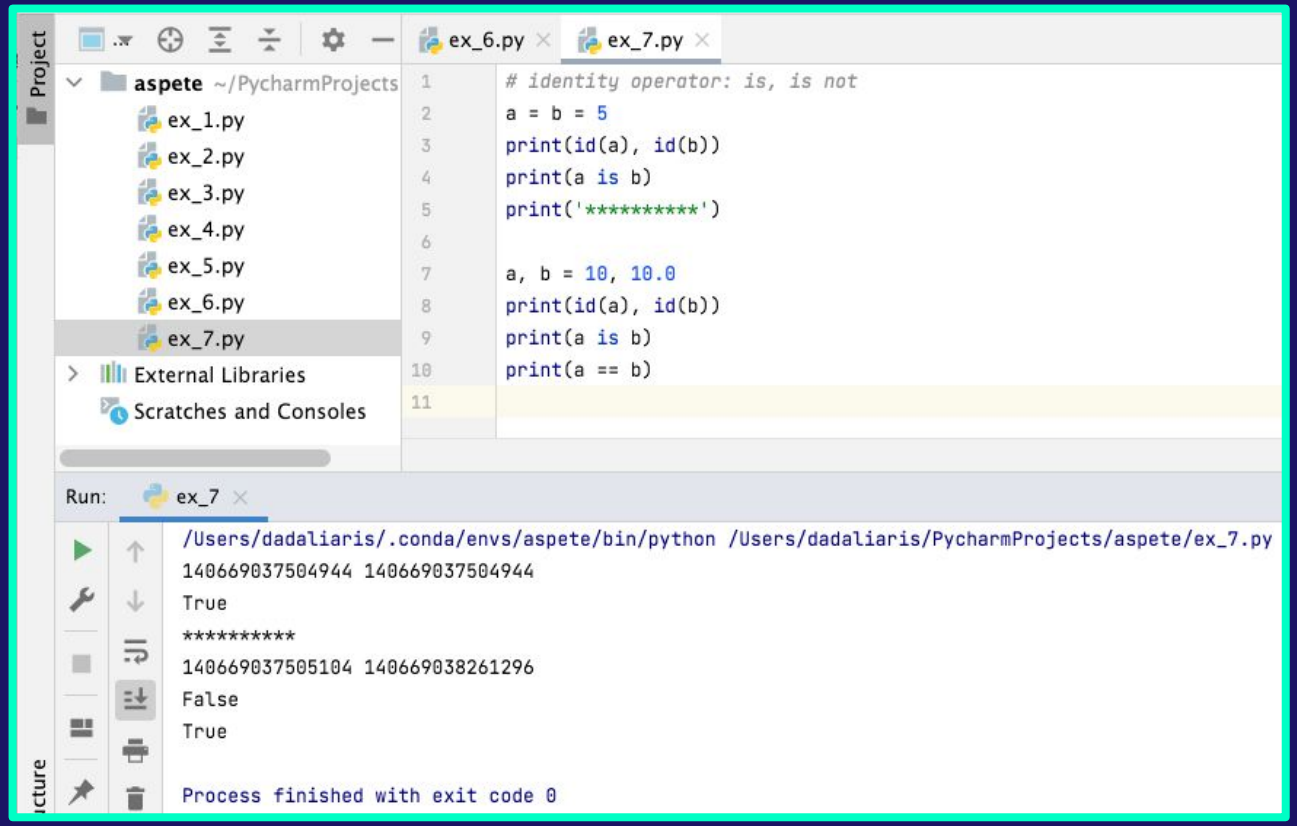
Flow Control / Conditional Statements (3)

- Operators:
  - lambda
  - If-else
  - or
  - and
  - not
  - in, not in, is, is not, <. <=, >, >=, !=, ==
  - |
  - ^
  - &
  - <<, >>
  - +, -
  - *, /, //, %
  - **
  - ()

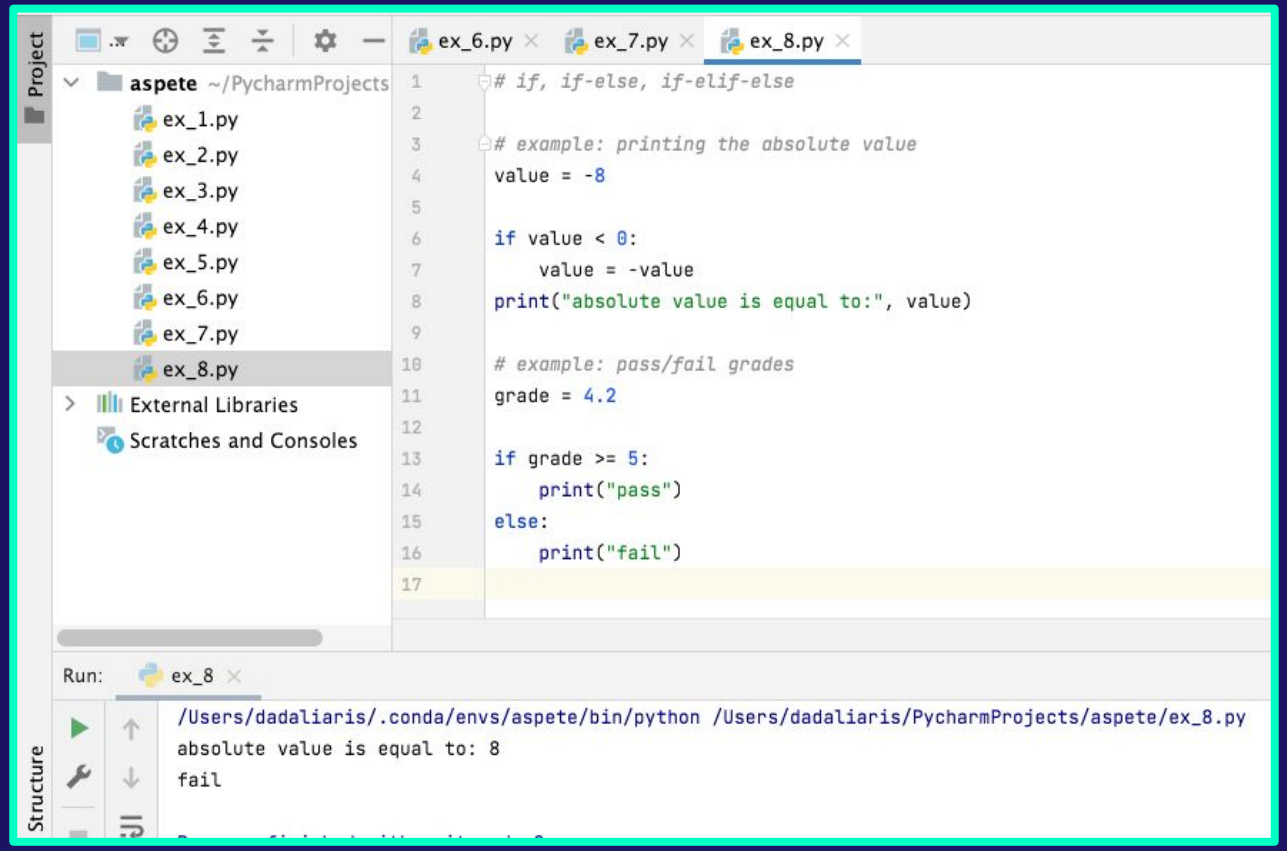
Chain comparison:

- a < b < c
- a < b > c
- a > b < c

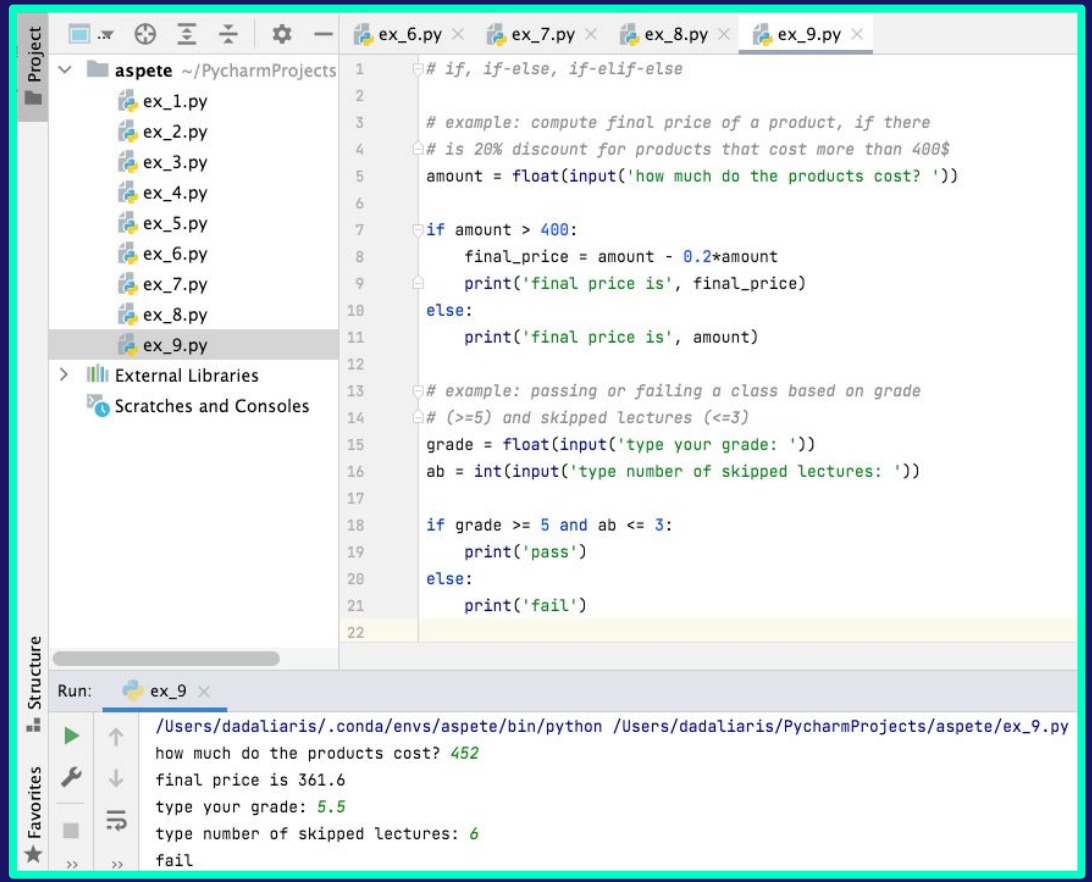
Flow Control / Conditional Statements (4)



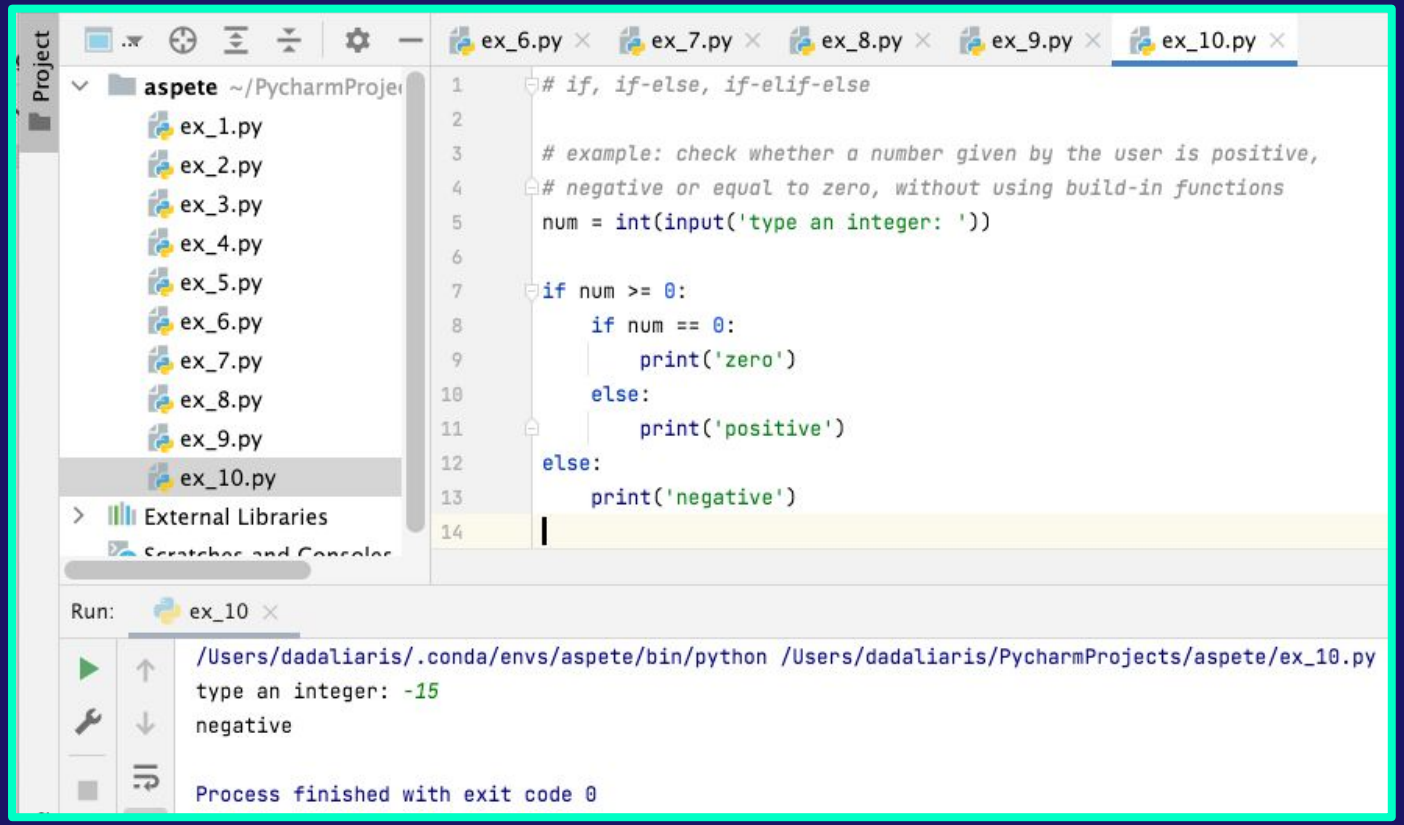
Flow Control / Conditional Statements (5)



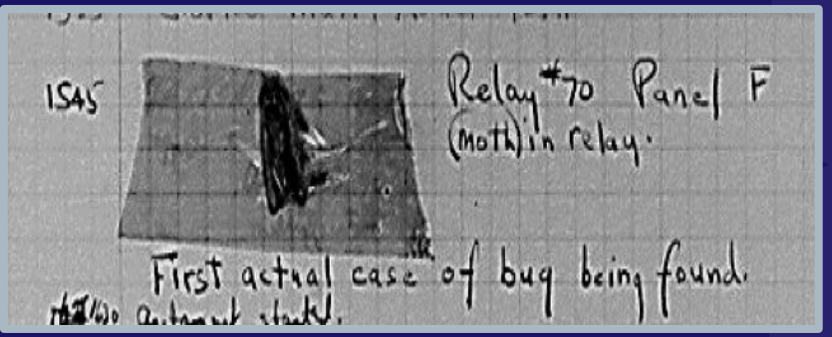
Flow Control / Conditional Statements (6)



Flow Control / Conditional Statements (7)

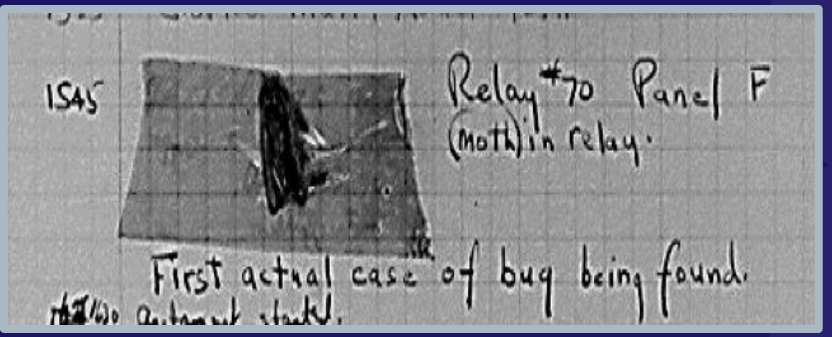


Debugging (1)



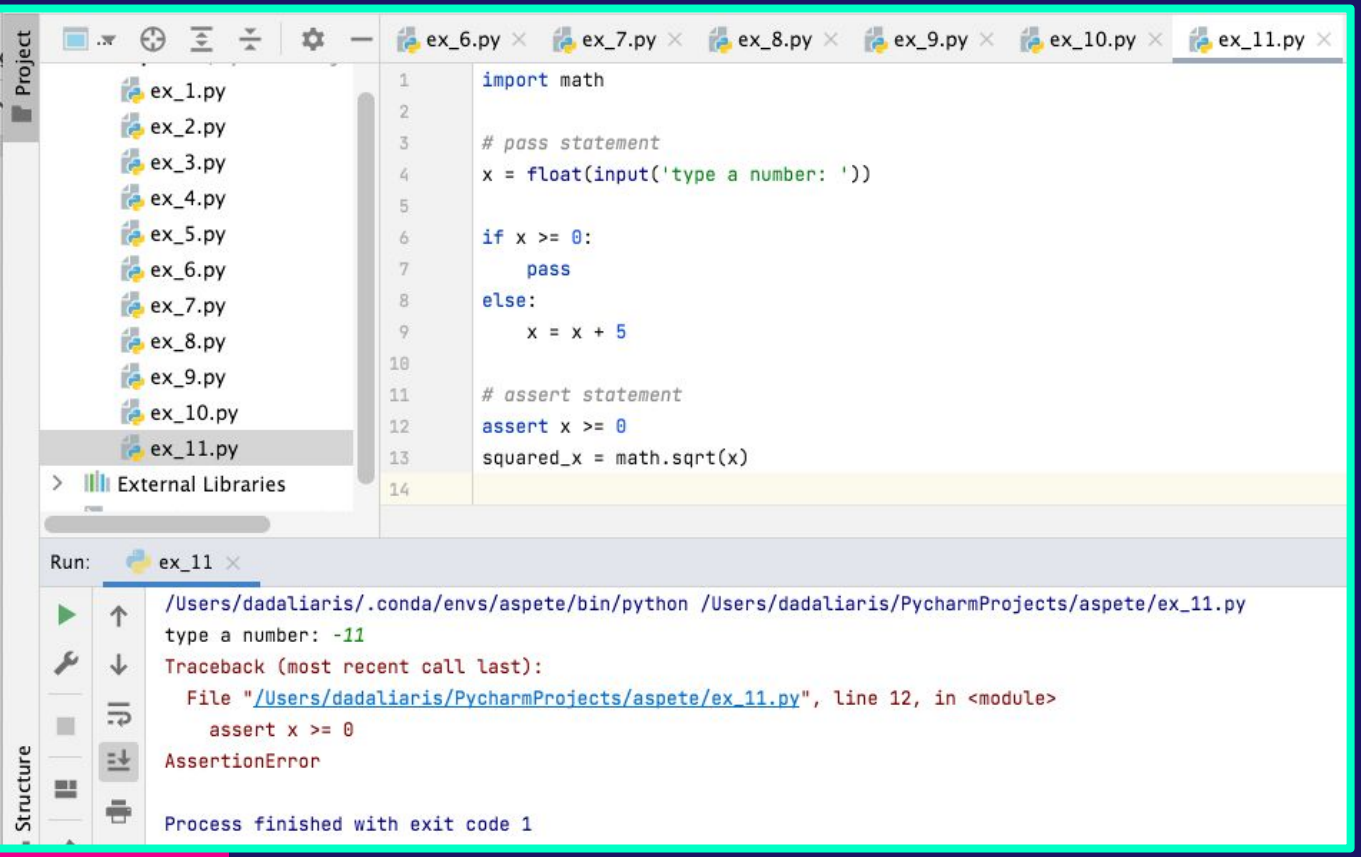
- Critical skill in programming (you must learn how to do it and do it well)
- No single way to do it
- Biggest obstacle: “my code is right and the computer is wrong”
- Your best “friend”: the debugger
- Two approaches depending on the program’s misbehavior:
  - Program runs without crashing: use the debugger to watch the program’s execution, stepping through the program one statement at a time.
  - Program is crashing: start program execution under the debugger, the debugger will automatically get control at the point of the crash, where you can examine variable values at that point.
- Why is it called a "bug": In the 1940s, in a U.S. Navy computer project using one of the first general-purpose computers (Mark I), a programmer found that a problem was due to a dead moth stuck in a relay in the machine. Removing the bug allowed the computer to run as programmed.

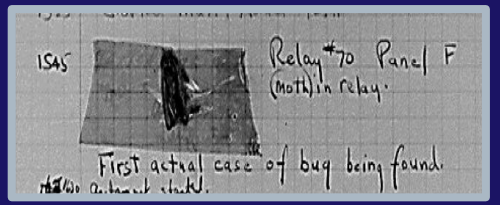
Debugging (2)



- "pass" statement
  - a null operation, when it is executed, nothing happens
  - useful as a placeholder when a statement is required syntactically, but no code needs to be executed
- "assert" statement:
  - statement has a condition or expression which is supposed to be always true. If the condition is false assert halts the program and gives an AssertionError
- Exceptions:
  - More on this later. . . .

Debugging (3)


Project
ex_1.py
ex_2.py
ex_3.py
ex_4.py
ex_5.py
ex_6.py
ex_7.py
ex_8.py
ex_9.py
ex_10.py
ex_11.py
External Libraries
ex_6.py
ex_7.py
ex_8.py
ex_9.py
ex_10.py
ex_11.py
1 import math
2
3 # pass statement
4 x = float(input('type a number: '))
5
6 if x >= 0:
7 pass
8 else:
9 x = x + 5
10
11 # assert statement
12 assert x >= 0
13 squared_x = math.sqrt(x)
14
Run:
ex_11
/Users/dadaliaris/.conda/envs/aspete/bin/python /Users/dadaliaris/PycharmProjects/aspete/ex_11.py
type a number: -11
Traceback (most recent call last):
File "/Users/dadaliaris/PycharmProjects/aspete/ex_11.py", line 12, in <module>
assert x >= 0
AssertionError
Process finished with exit code 1
Structure


1545
Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.

Python Loops (1)

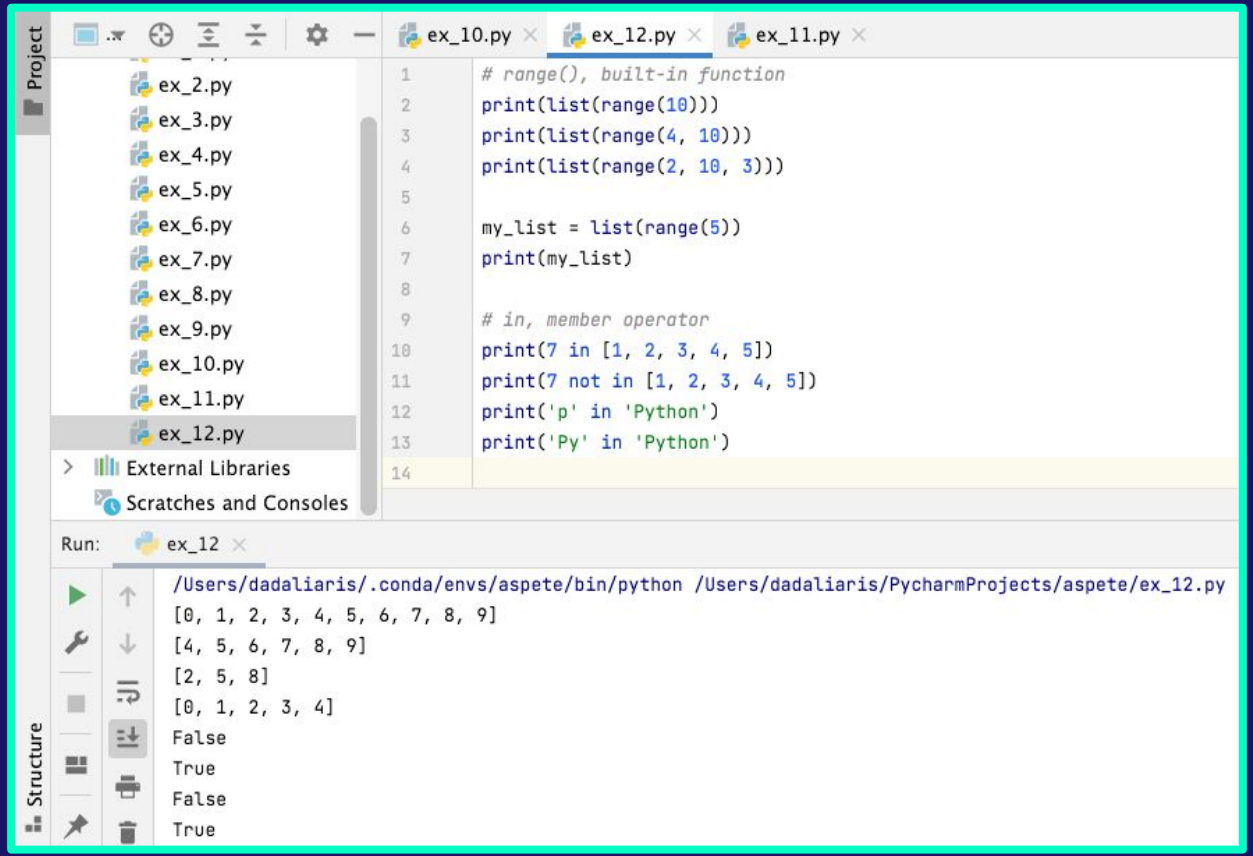
- Python has two basic loop constructs:
  - for
  - while

- Before we check their syntax and usage we should mention the “range“ built-in function and the “in“ member operator.

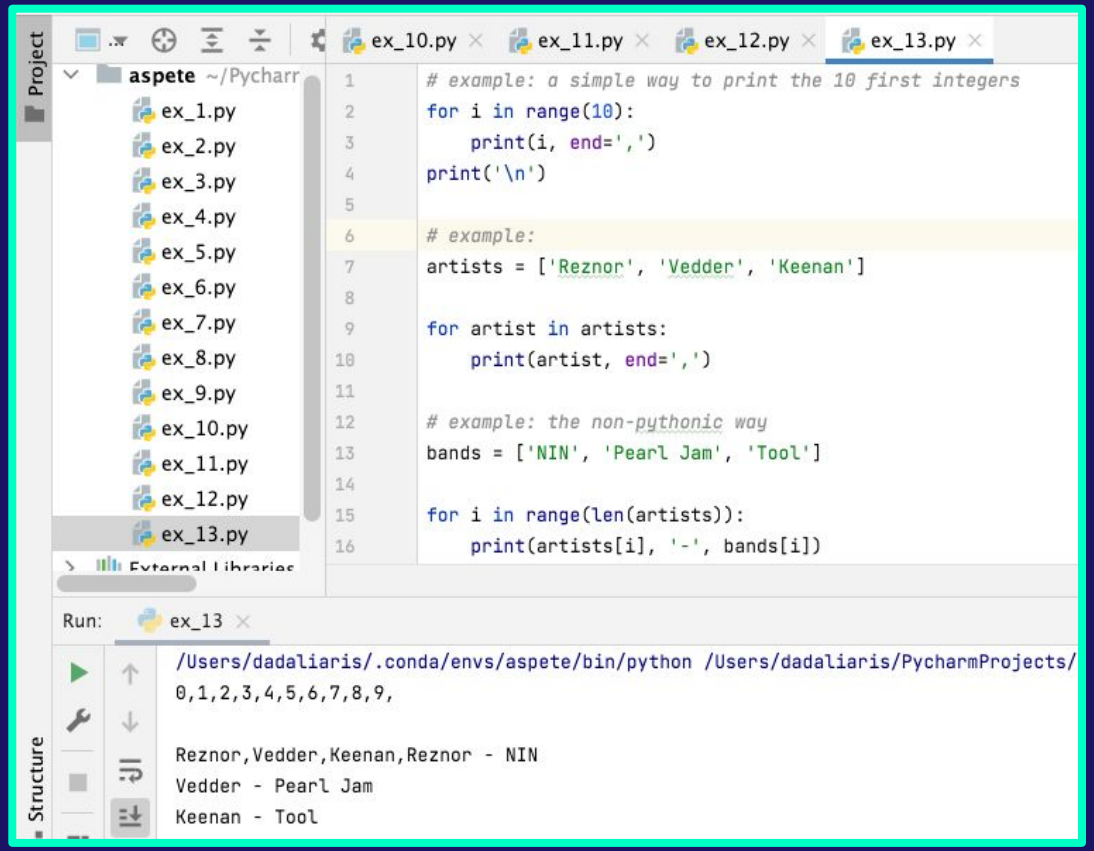
- “range“ is used to produce a sequence of numbers
- range(start, finish, step)

- “in“ is used for checking the existence of a value in a sequence of values

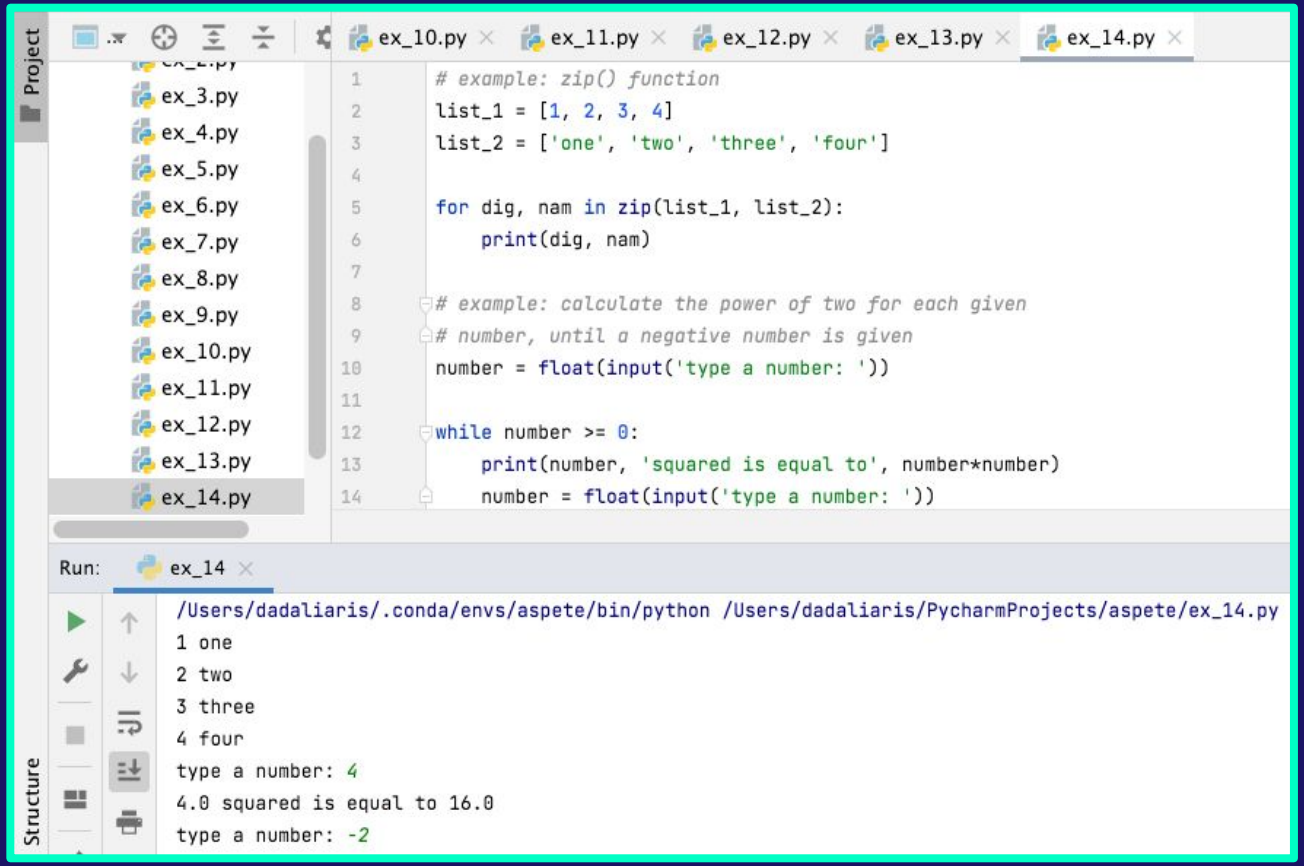
Python Loops (2)



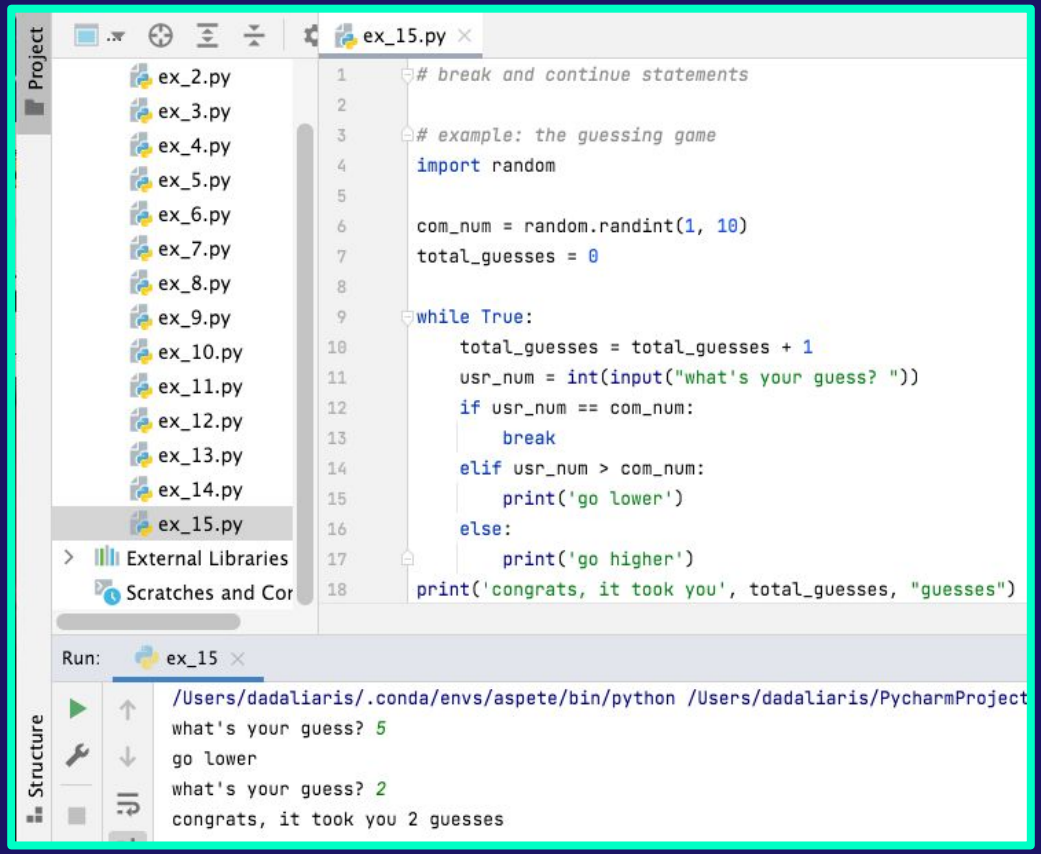
Python Loops (3)



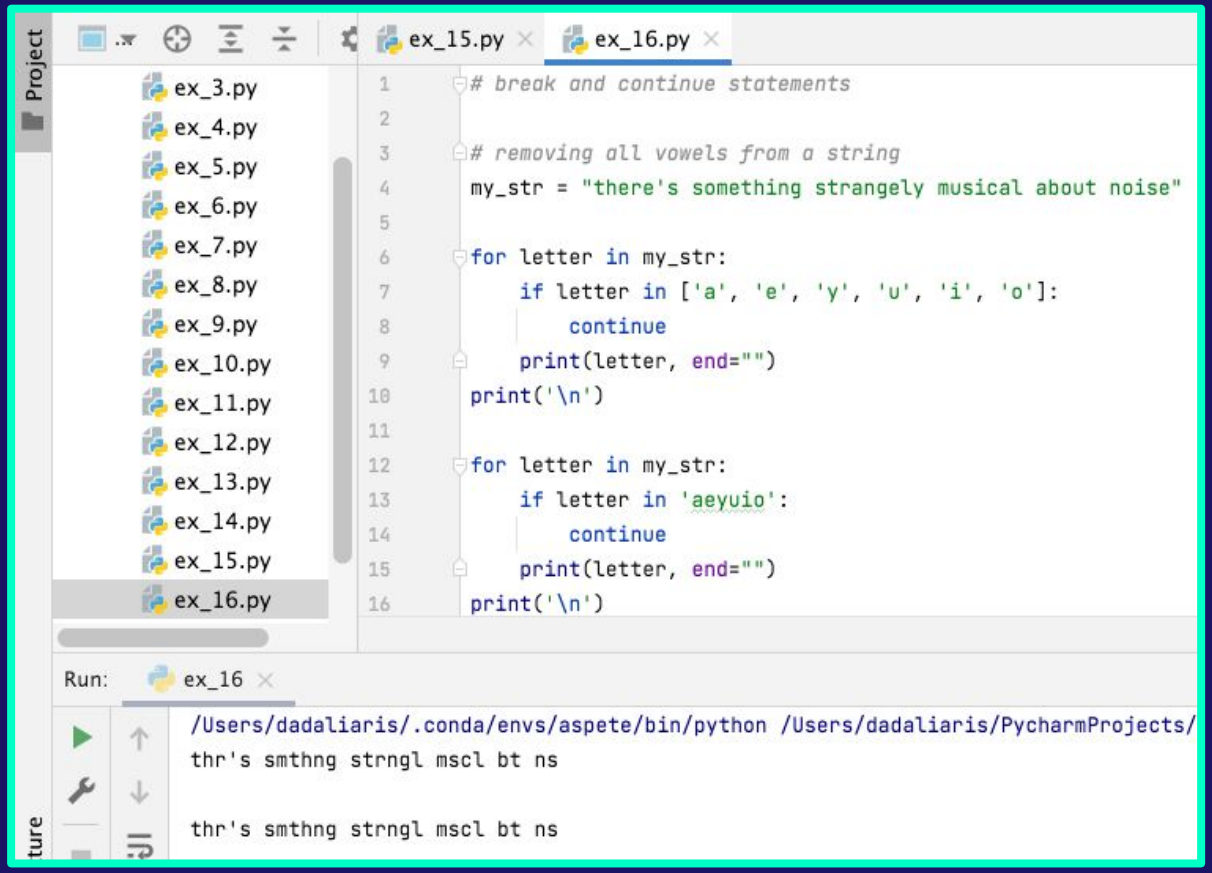
Python Loops (4)



Python Loops (5)



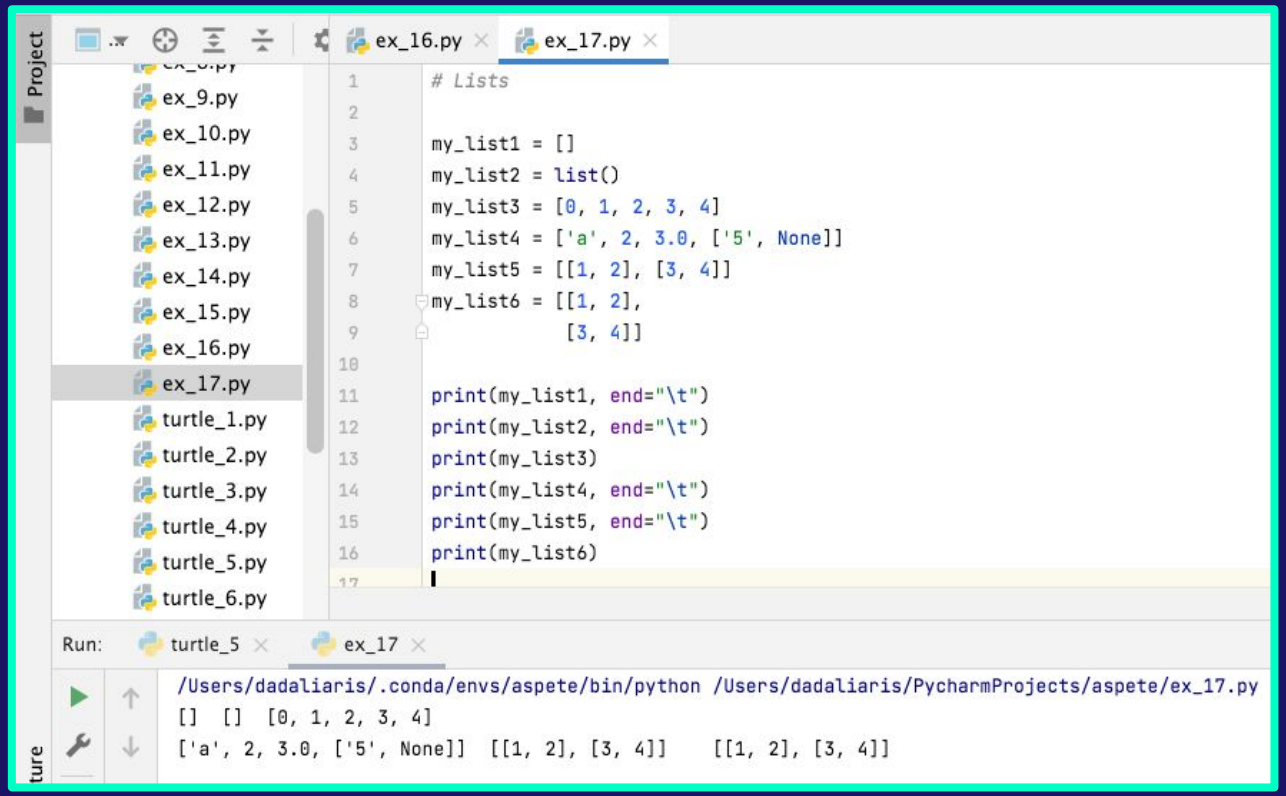
Python Loops (6)



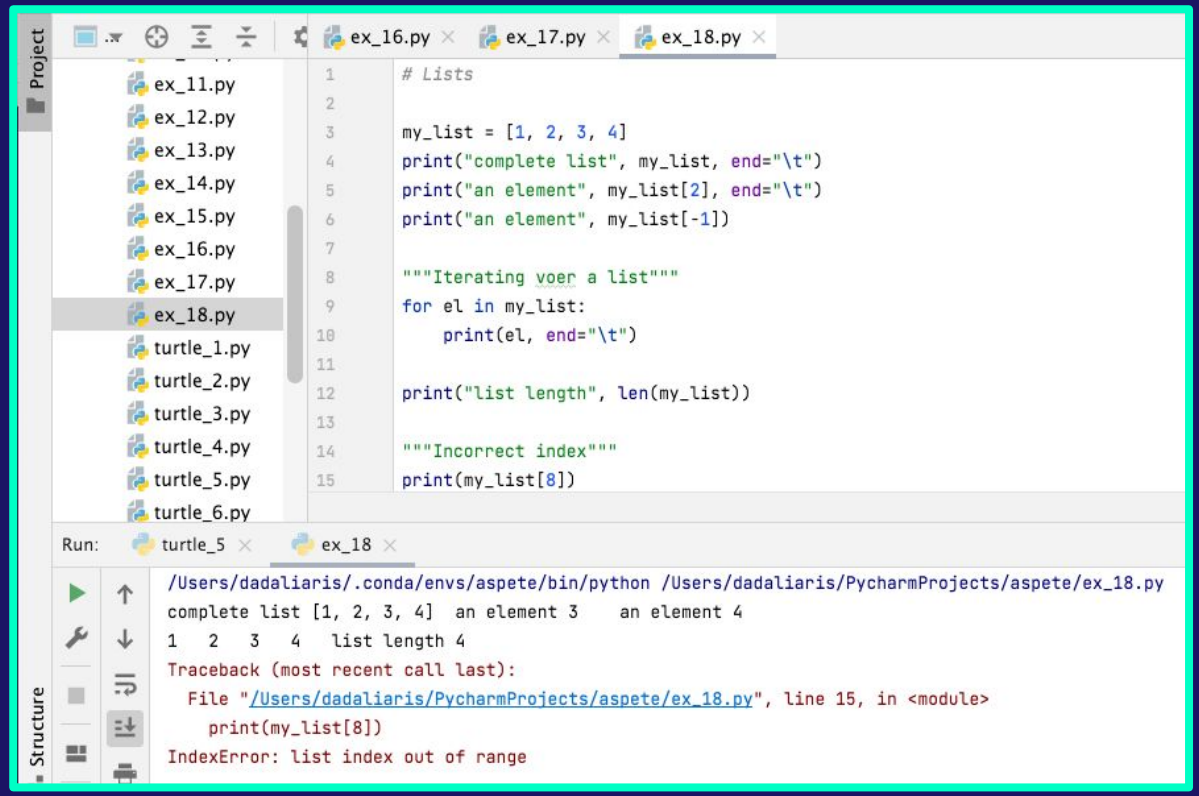
Python Lists (1)

- **List Definition & Characteristics:**
  - A collection or sequence of ordered objects.
  - Dynamic, its size can change at any time during the execution of a program.
  - Can contain objects of any size and any type.
  - Has built-in methods.
  - Is a mutable object that might contain immutable elements.

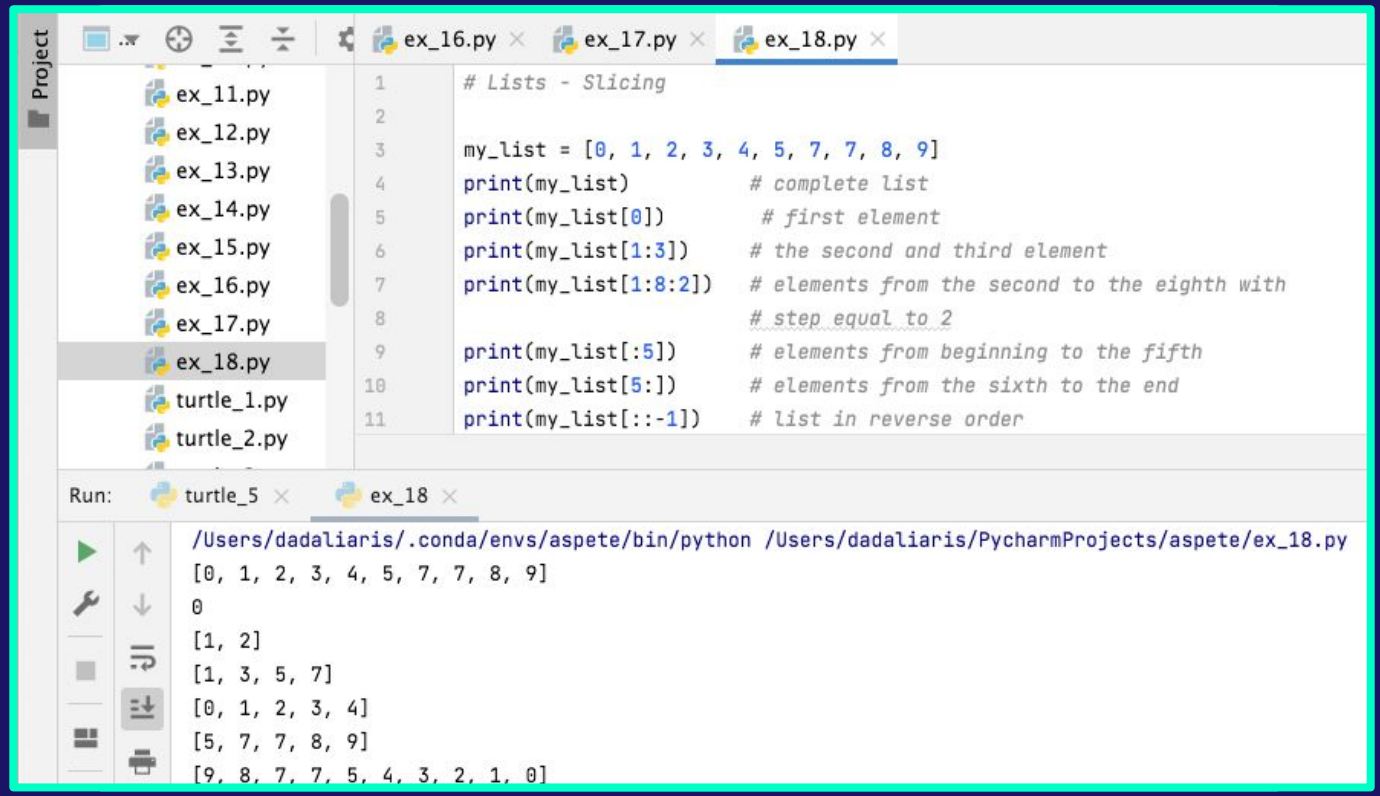
Python Lists (2)



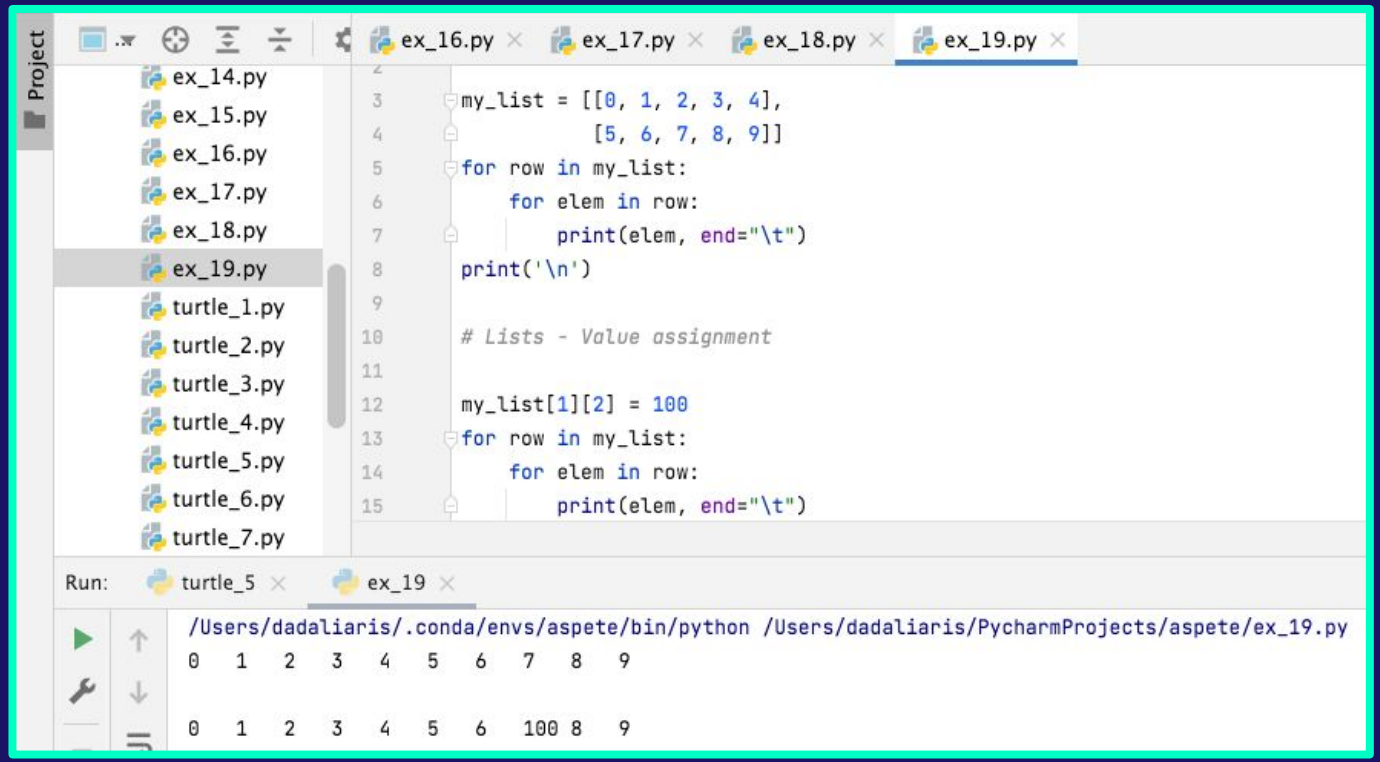
Python Lists (3)



Python Lists (4)



Python Lists (5)

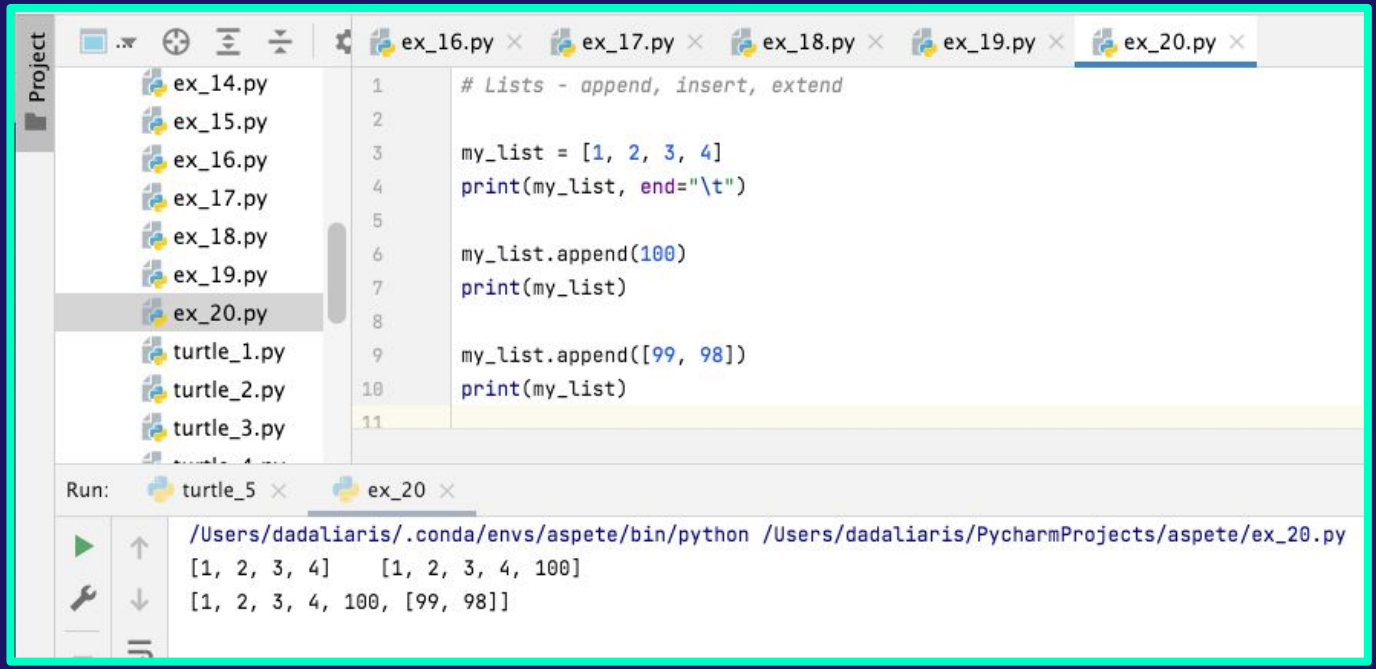


Python Lists (6)

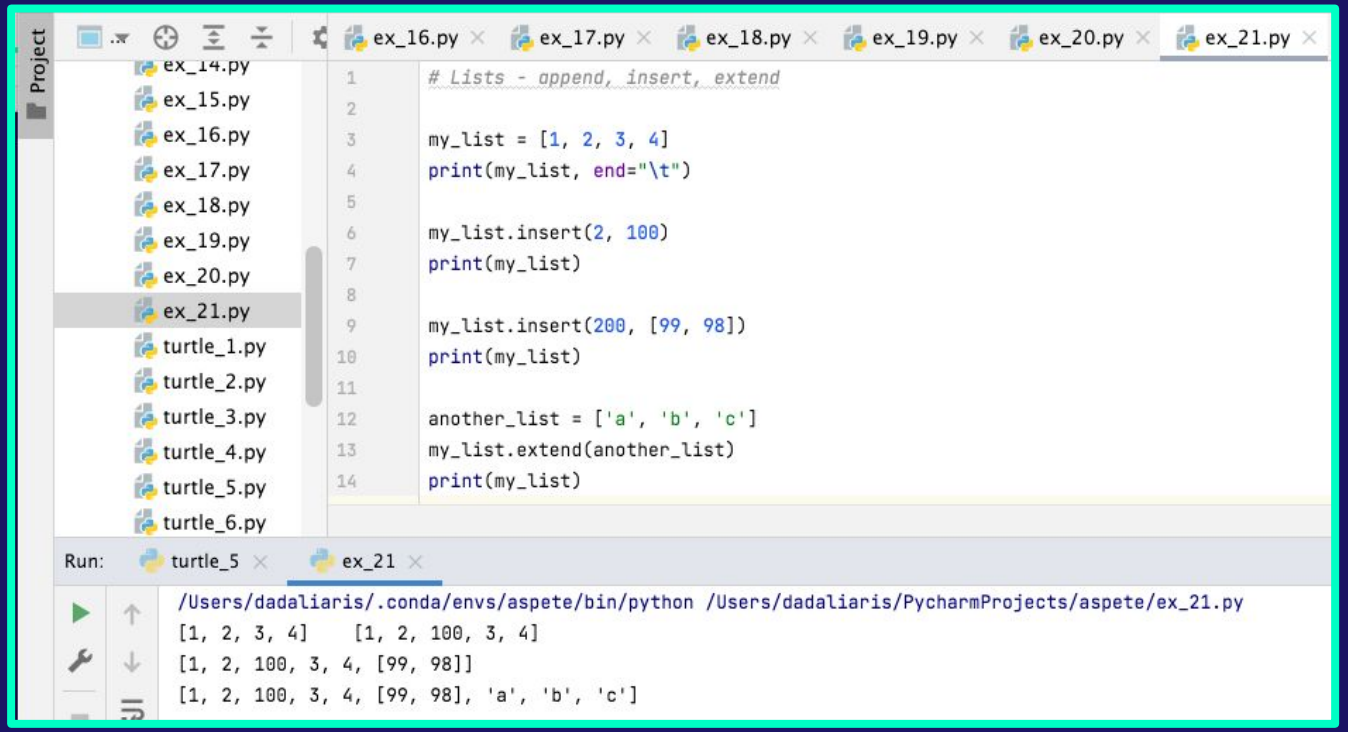
- There are three distinct ways to add new elements on a list:
  - “append”: adds a new element at the end of a list (in-place)
  - “Insert”: adds a new element at a specific index of a list
  - “extend”: adds a new sequence of elements at the end of a list

- There are four distinct ways to delete a list or elements of a list:
  - “del”
  - “clear”
  - “remove”
  - “pop”

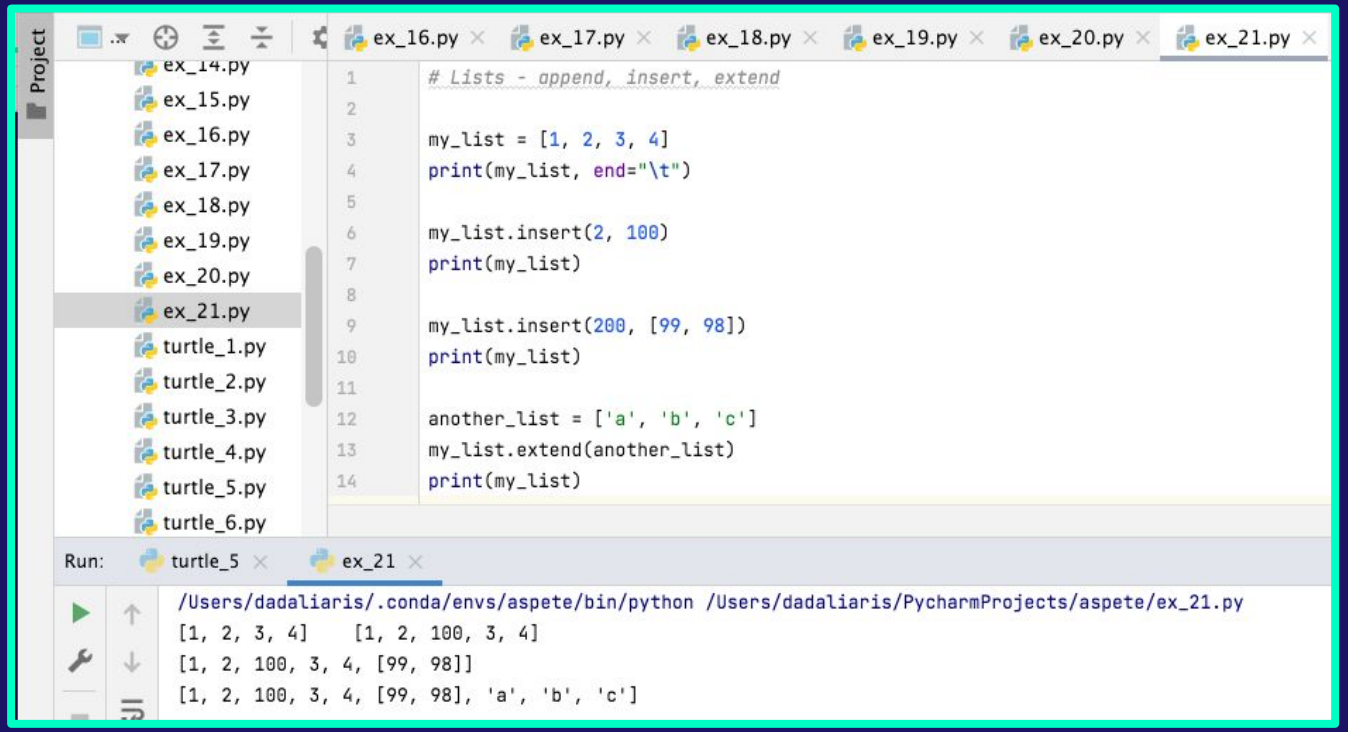
Python Lists (7)



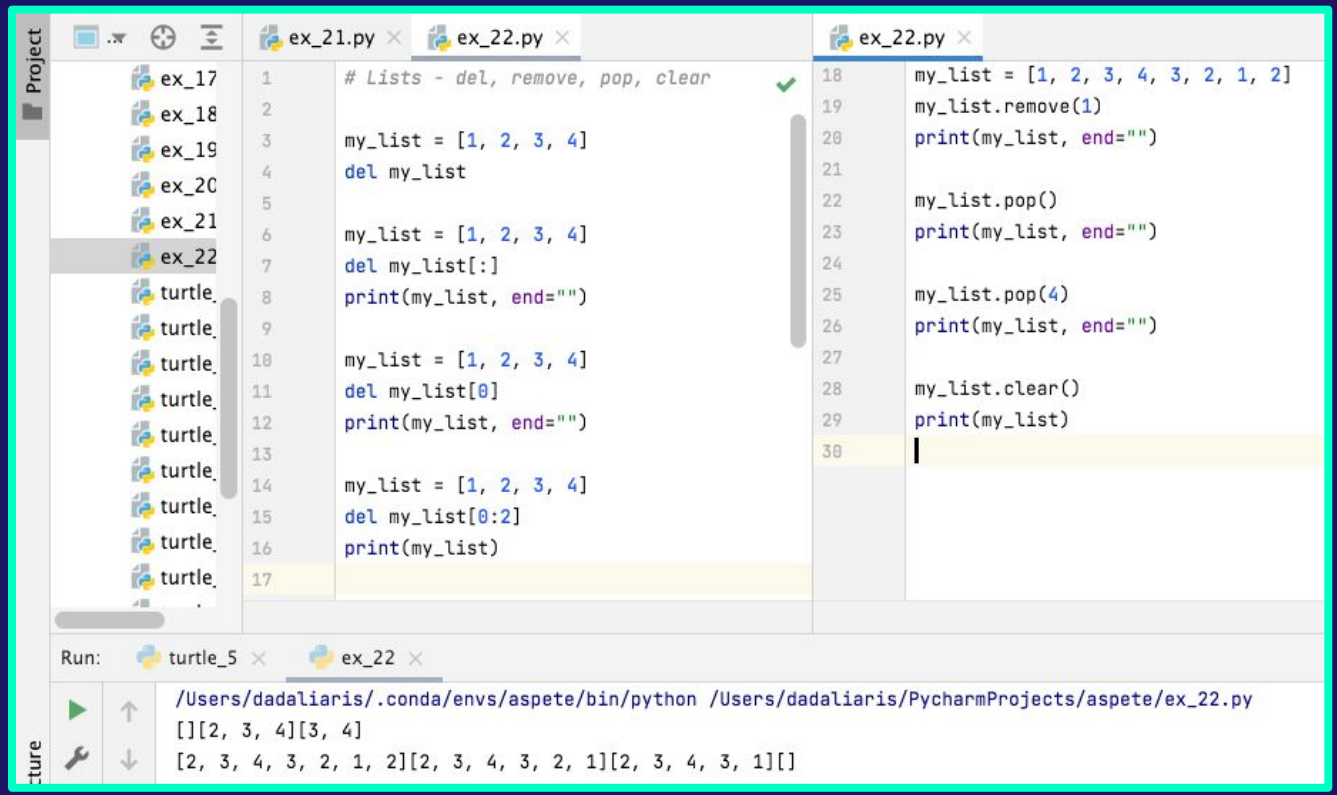
Python Lists (8)



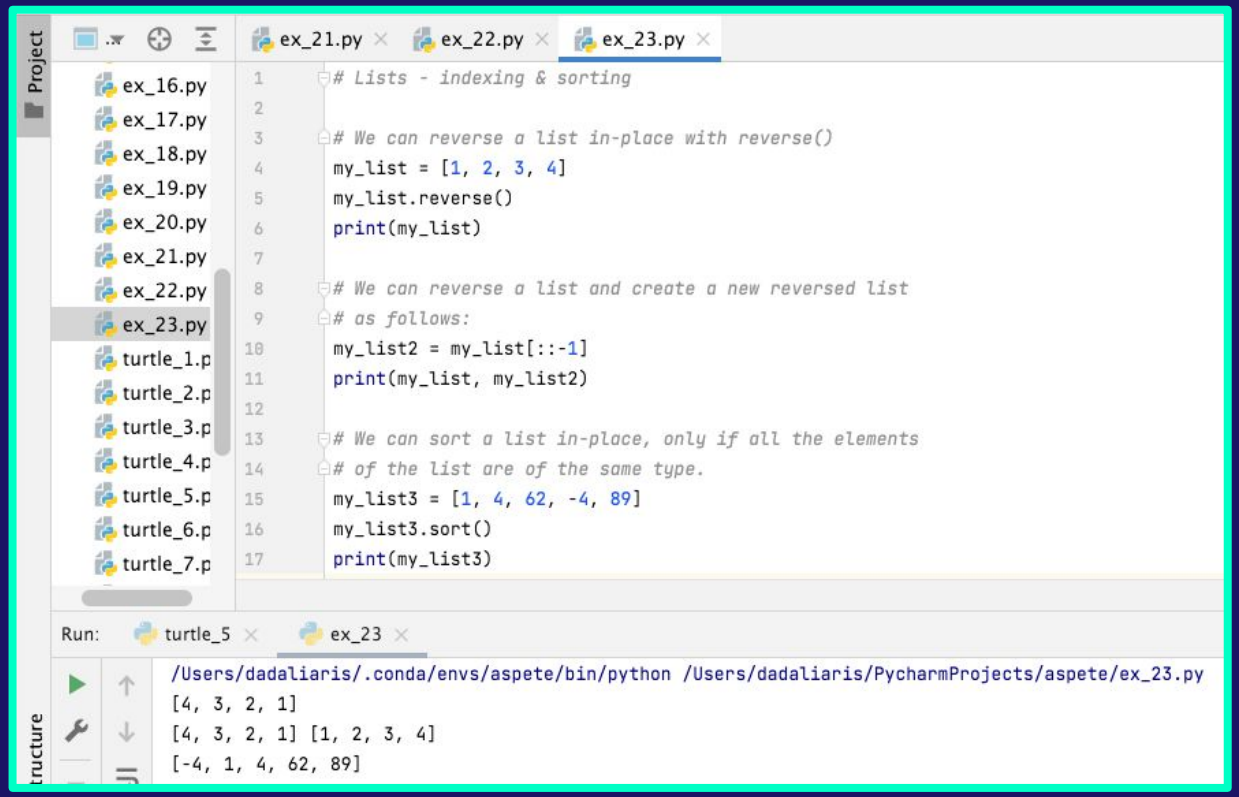
Python Lists (9)



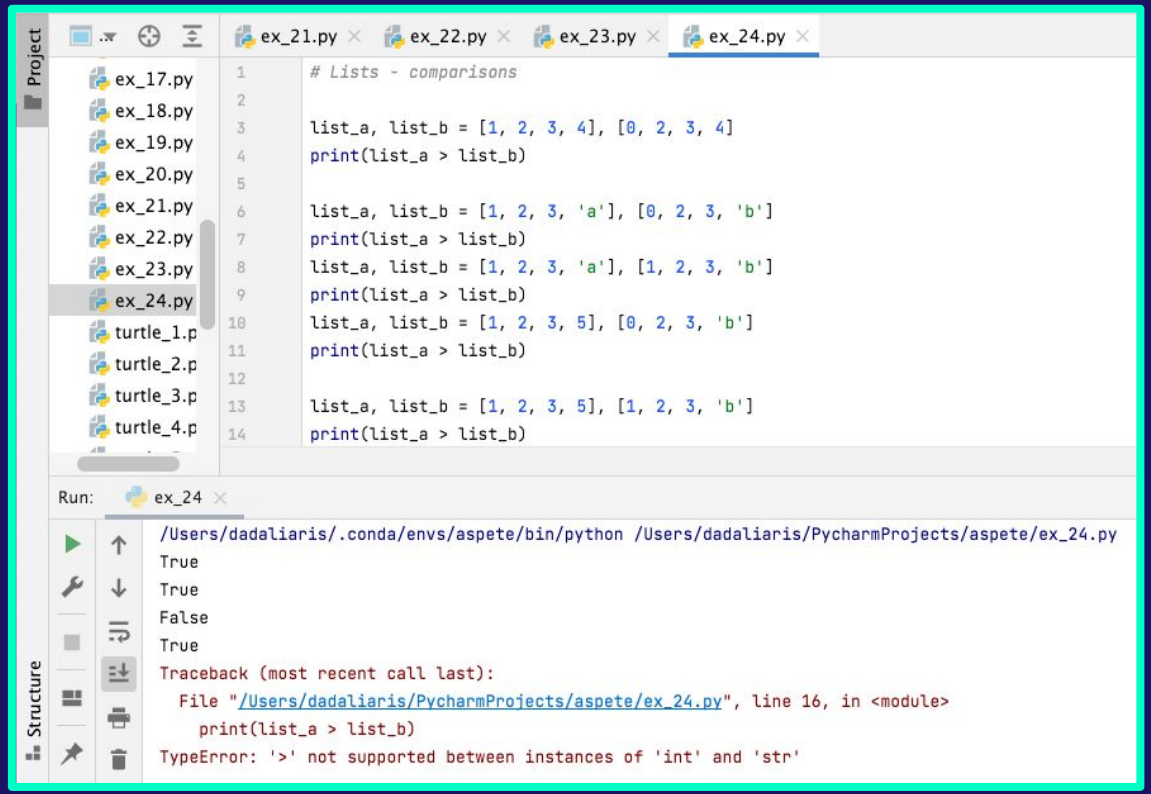
Python Lists (10)



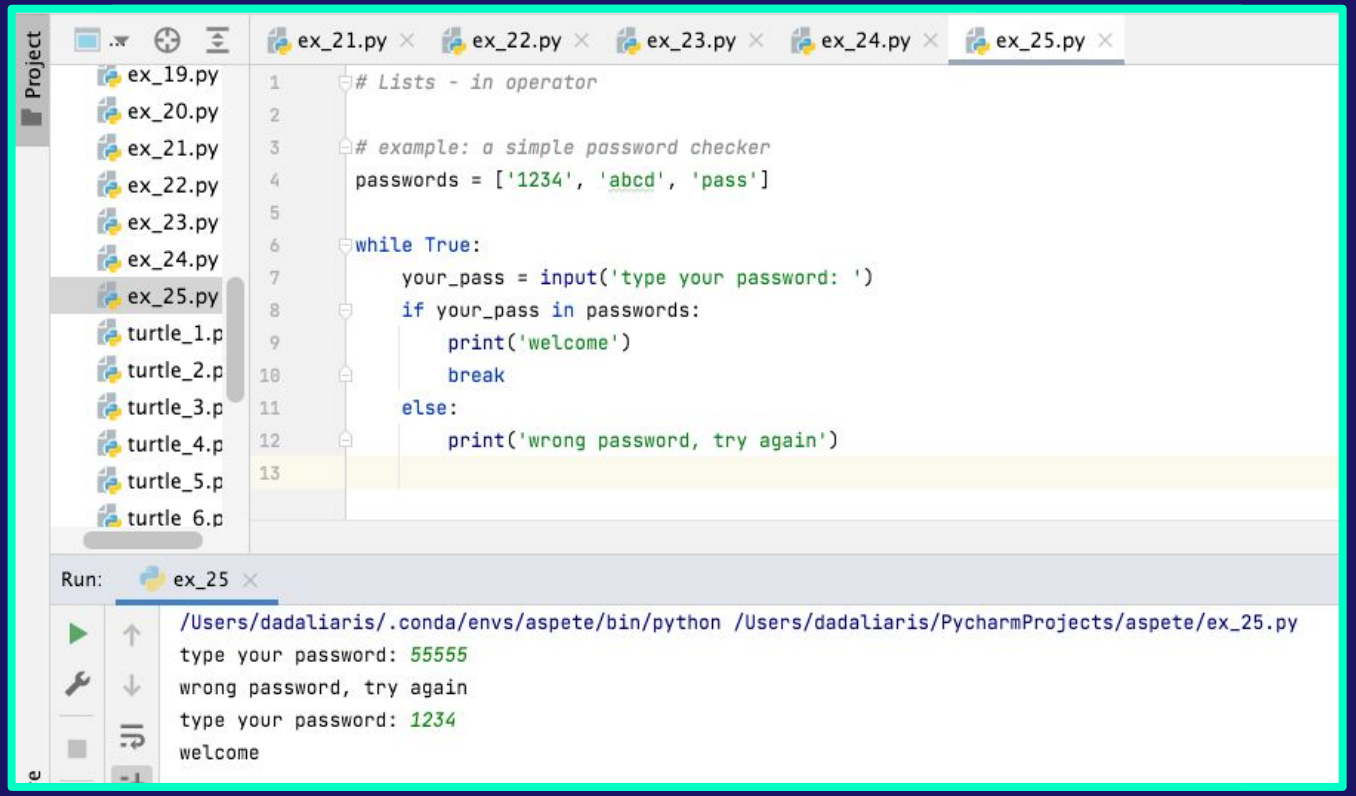
Python Lists (11)



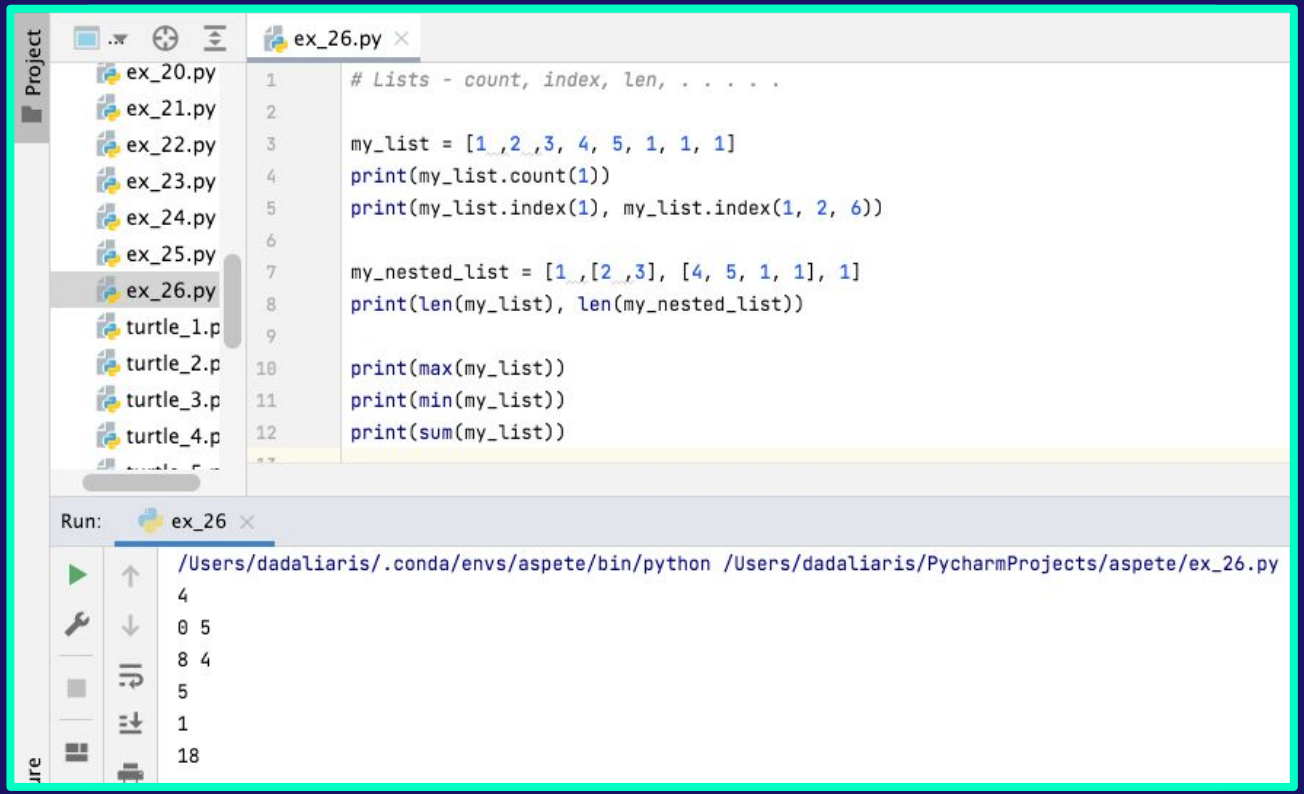
Python Lists (12)



Python Lists (13)



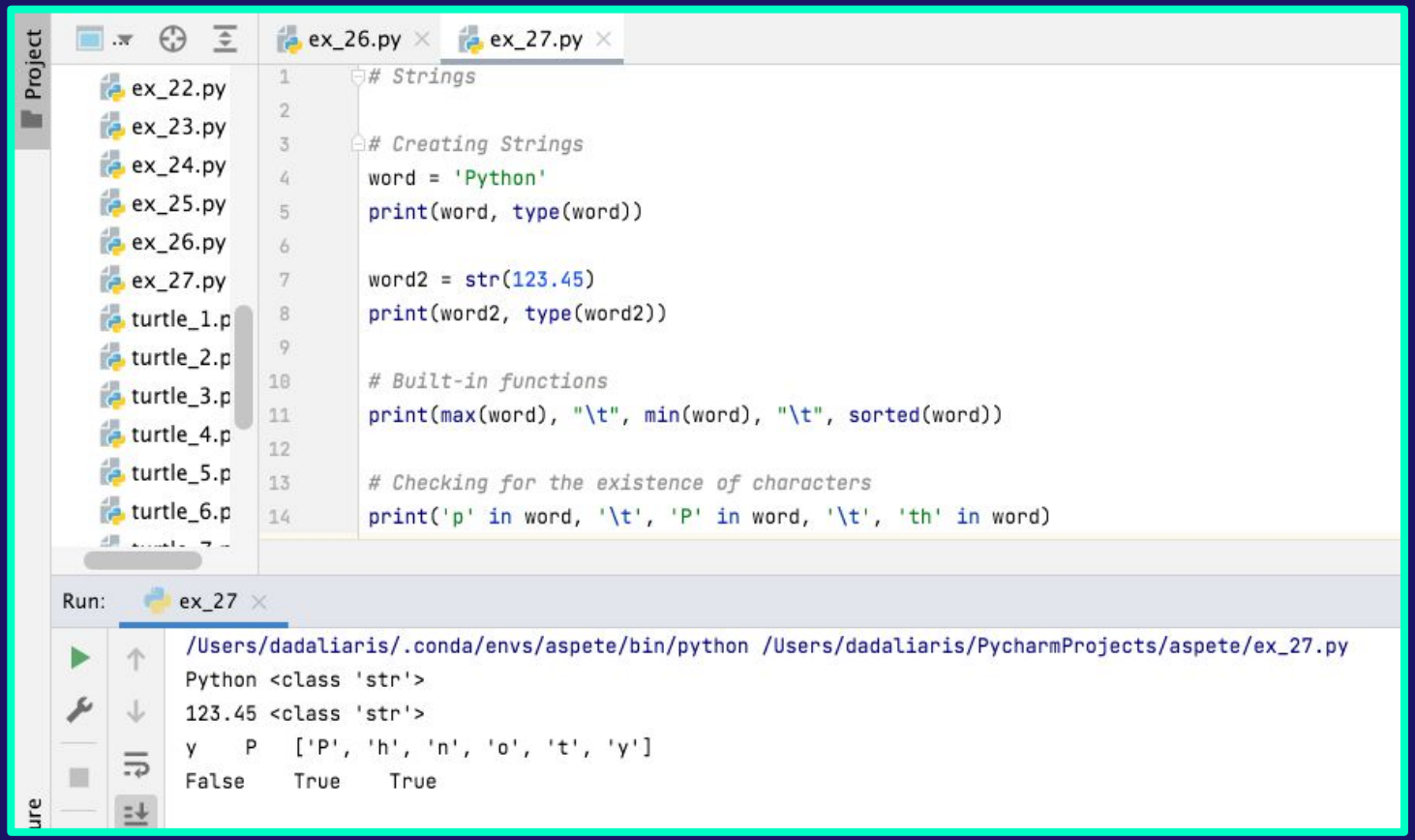
Python Lists (14)



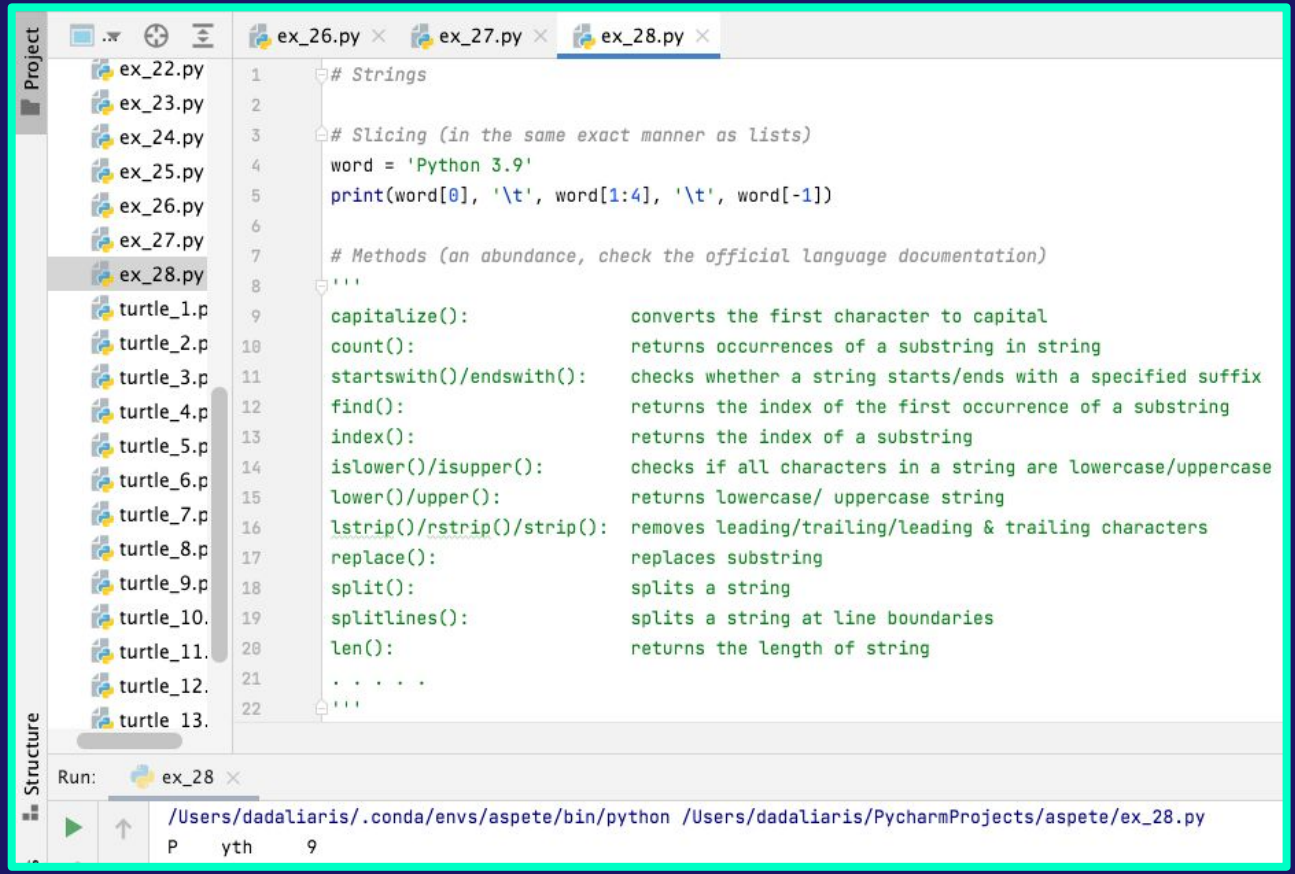
Python Strings (1)

- A string is an ordered and immutable sequence of characters.
- There is no specific type for characters in Python. A character is simply a string with length equal to one.
- Strings can be declared using single quotes (‘ ’), double quotes (“ ”) or triple single/double quotes (‘‘‘ ’’’/ “““ ”””).
- Triple quotes can be used for multi-line strings and should only be used for commenting our code(documentation/docstring).

Python Strings (2)



Python Strings (3)



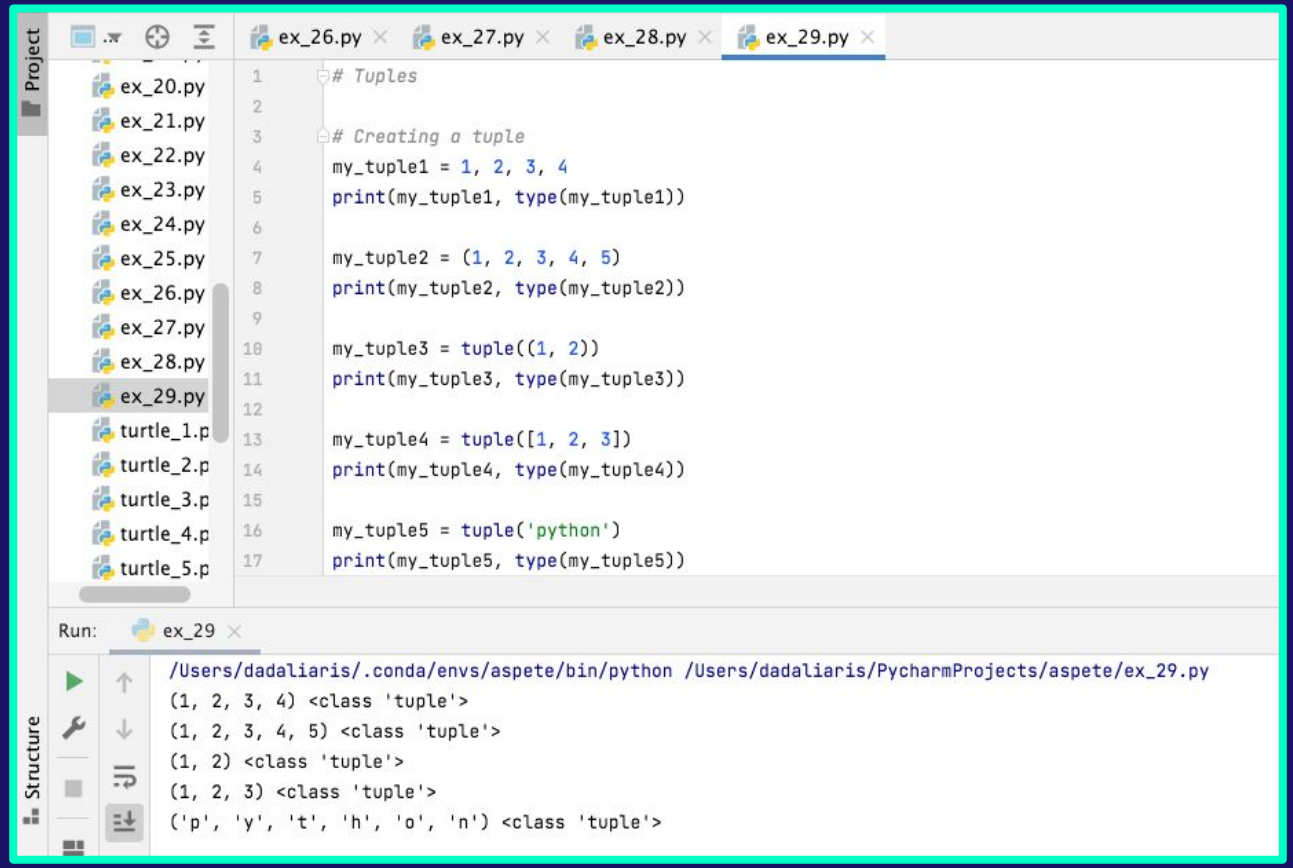
Python Tuples (1)

- A tuple is an immutable & iterable ordered sequence of element/objects.

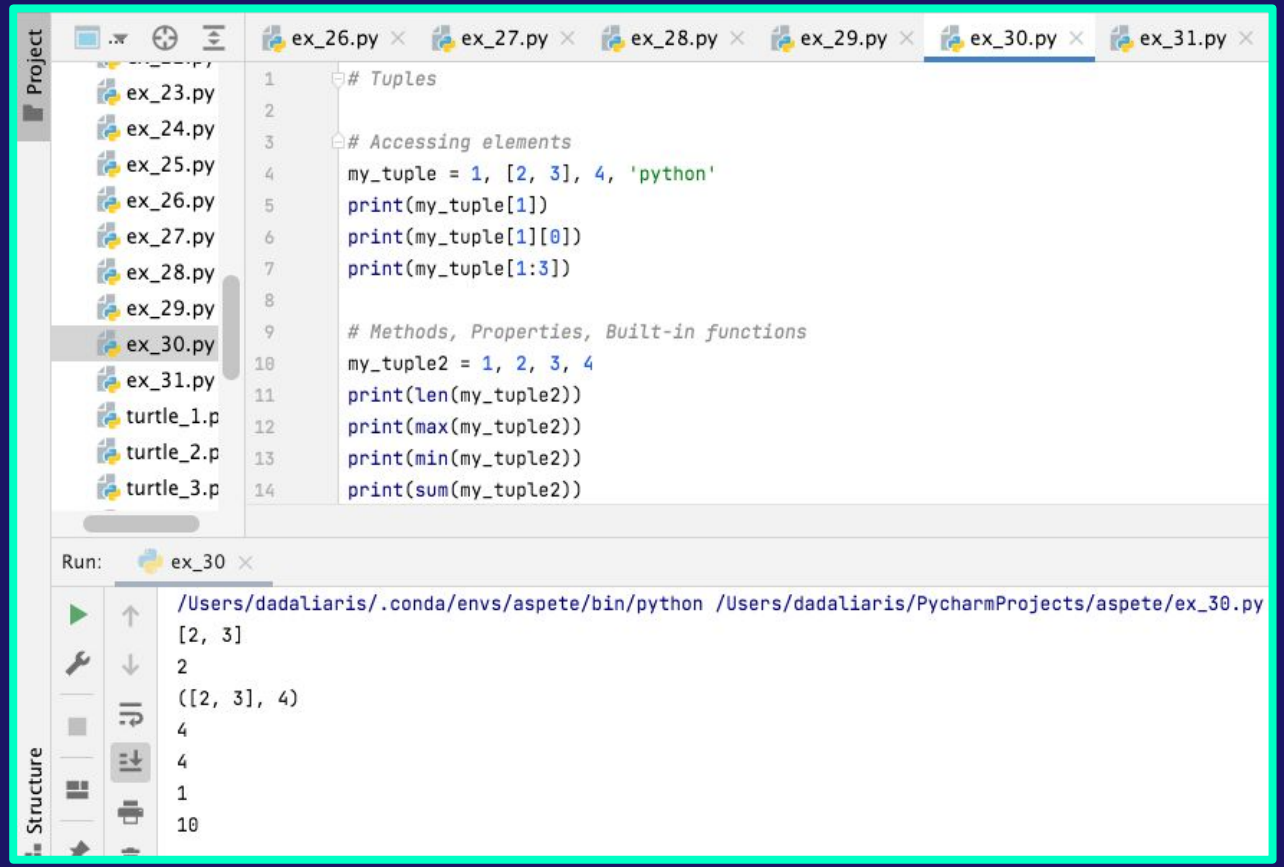
- Immutable:
    - We can delete a tuple
    - We cannot delete/modify its content

- Similar methods to a list, except those that might change its content.

Python Tuples (2)



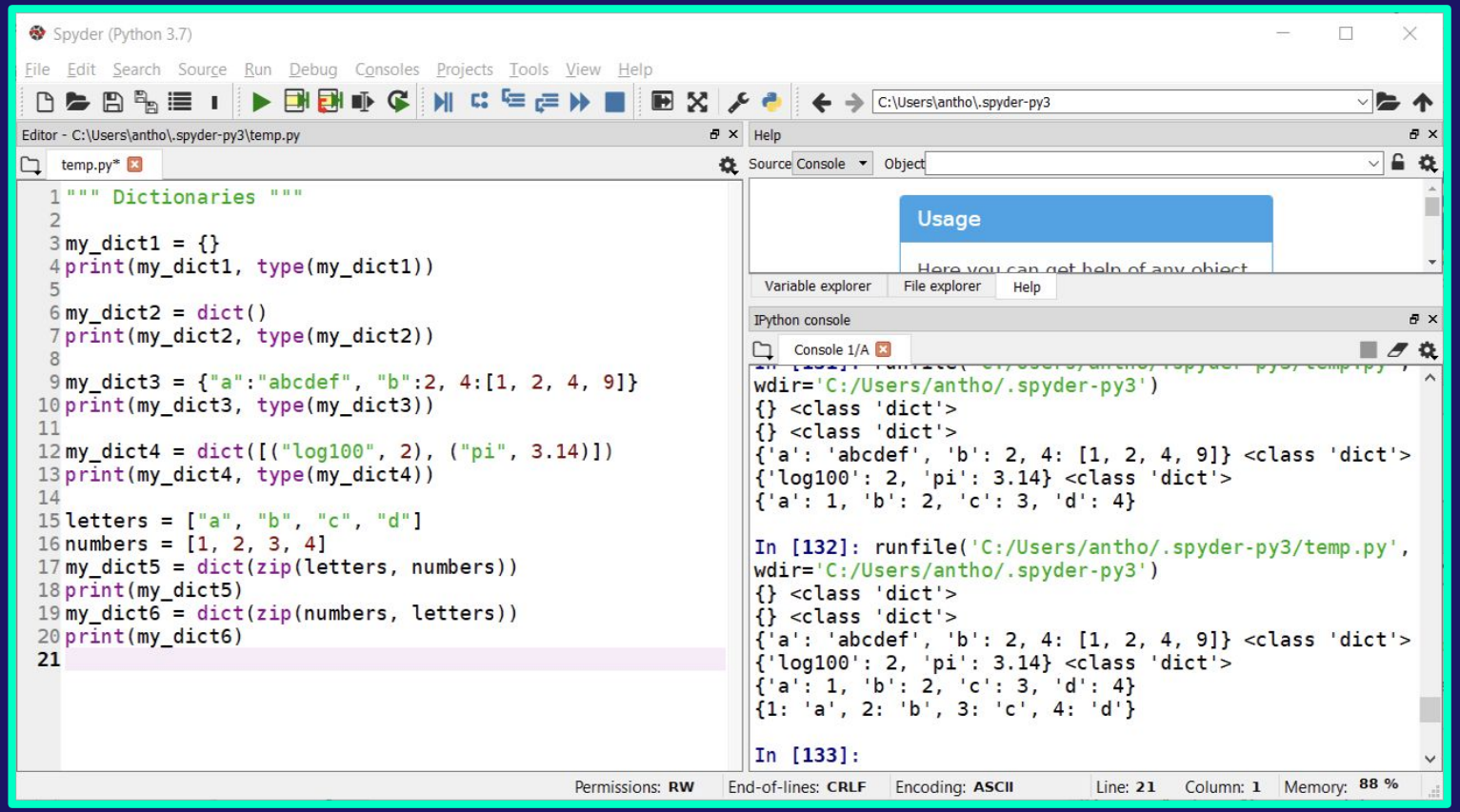
Python Tuples (3)



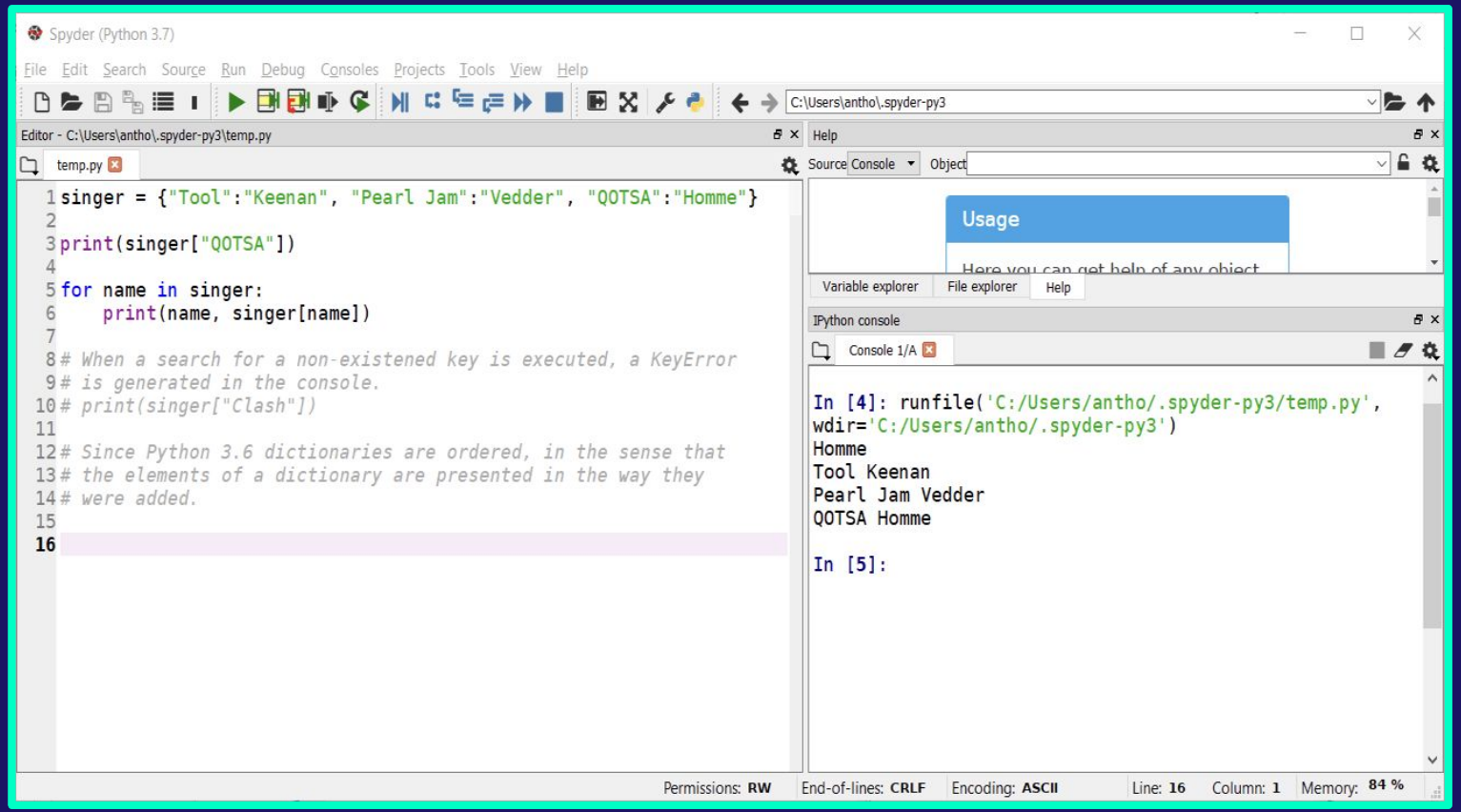
Python Dictionaries (1)

- A dictionary is mutable, iterable & unordered sequence of key-value pairs.
    - Keys must be unique
    - Key types must be immutable
    - Value can be of any type

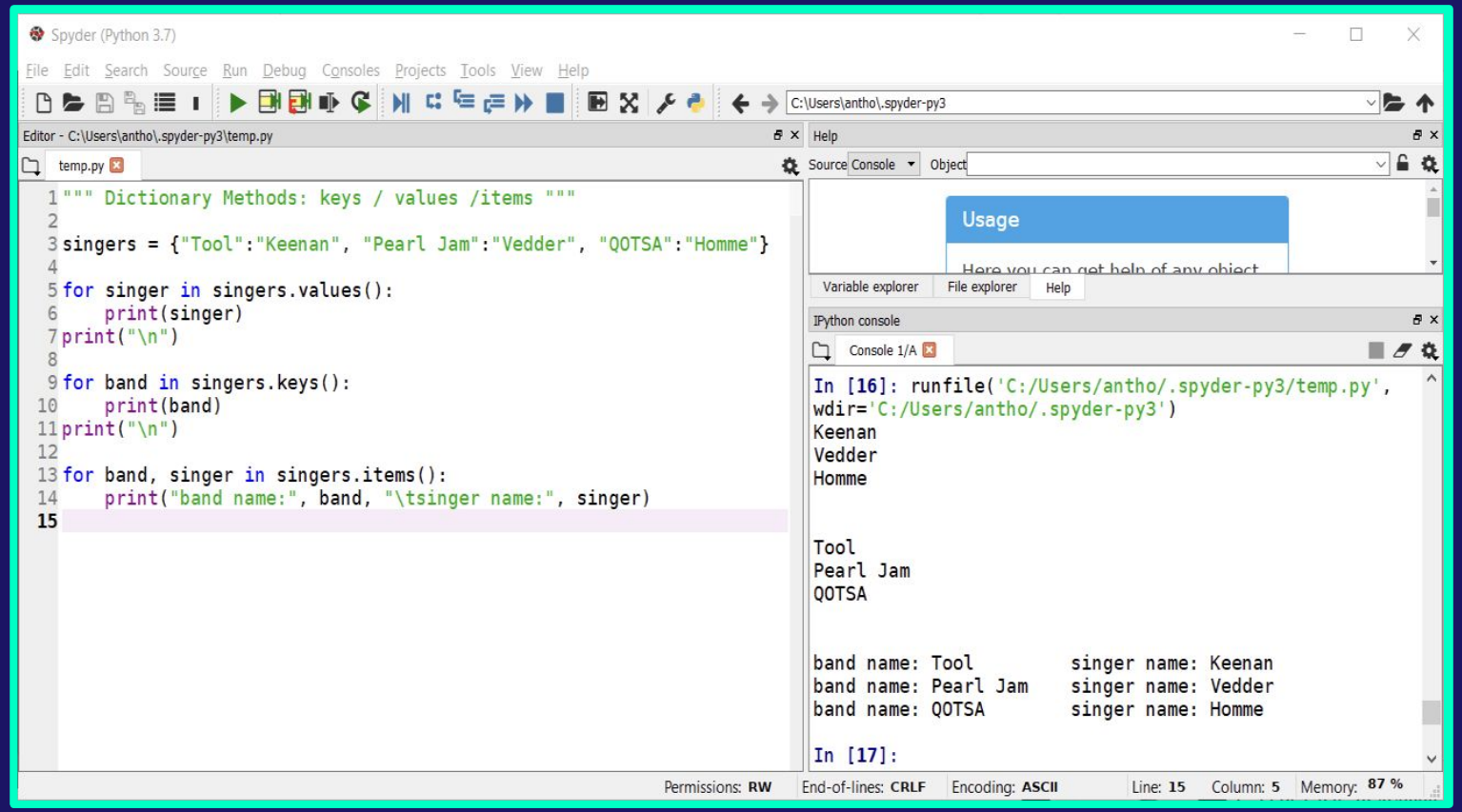
Python Dictionaries (2)



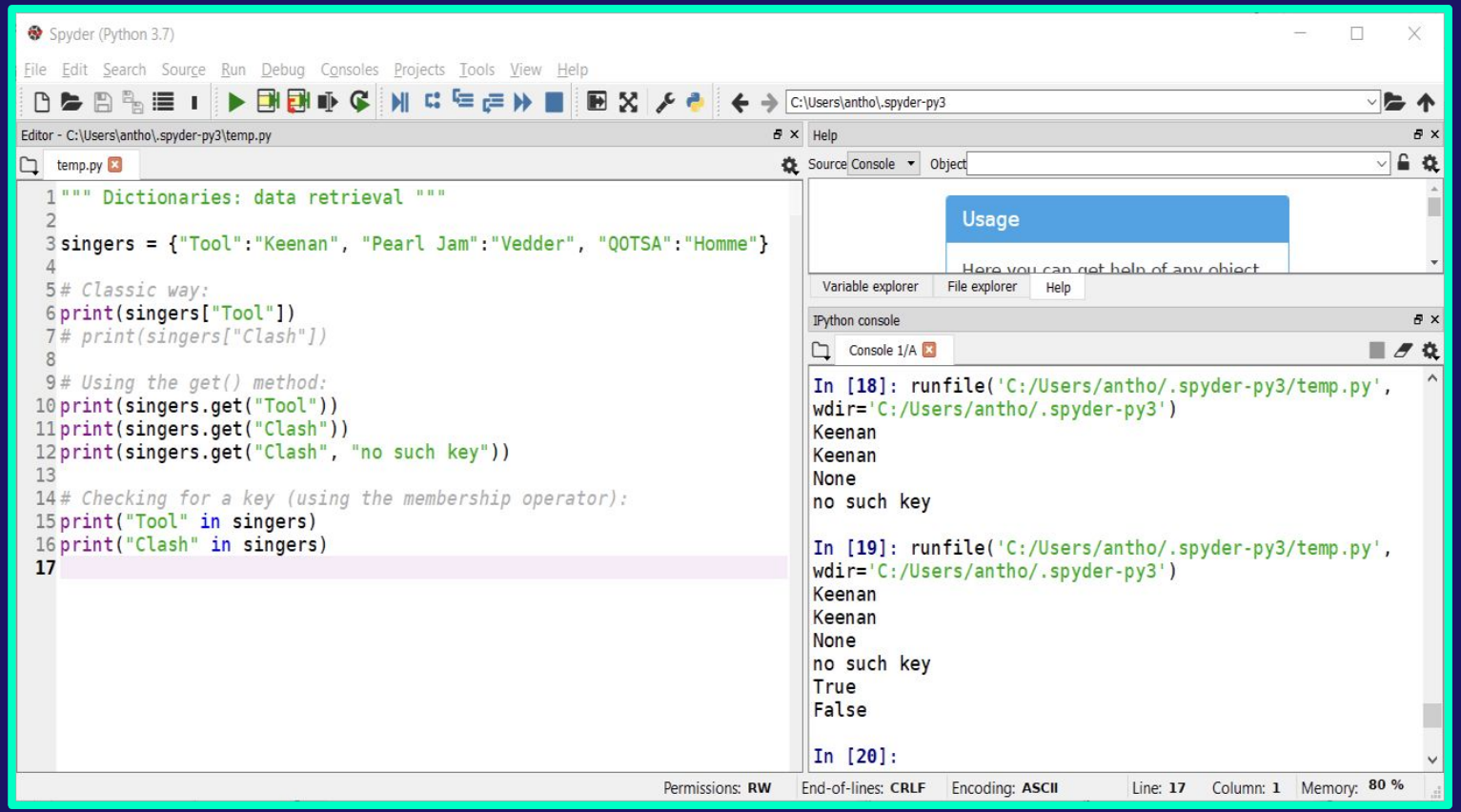
Python Dictionaries (3)



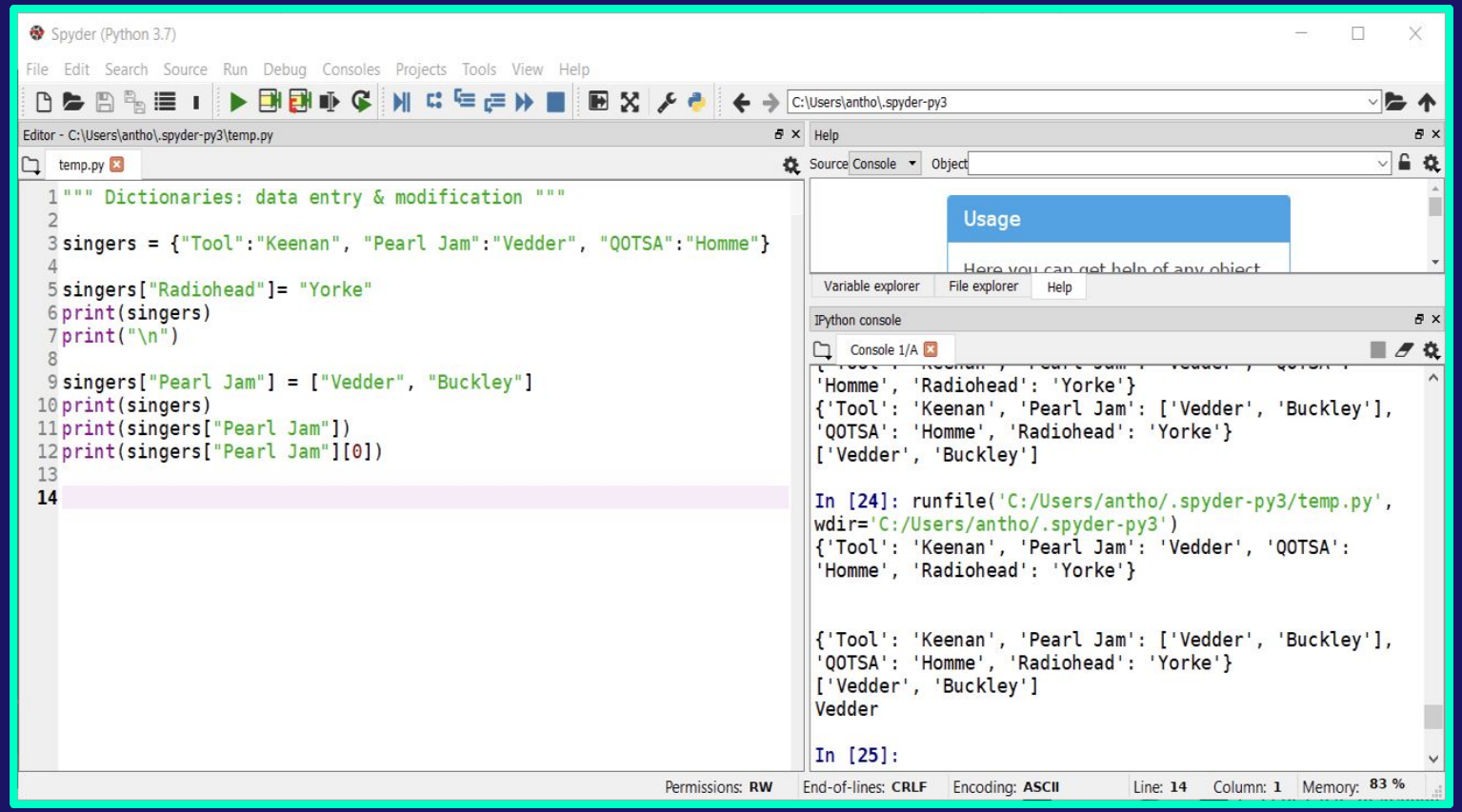
Python Dictionaries (4)



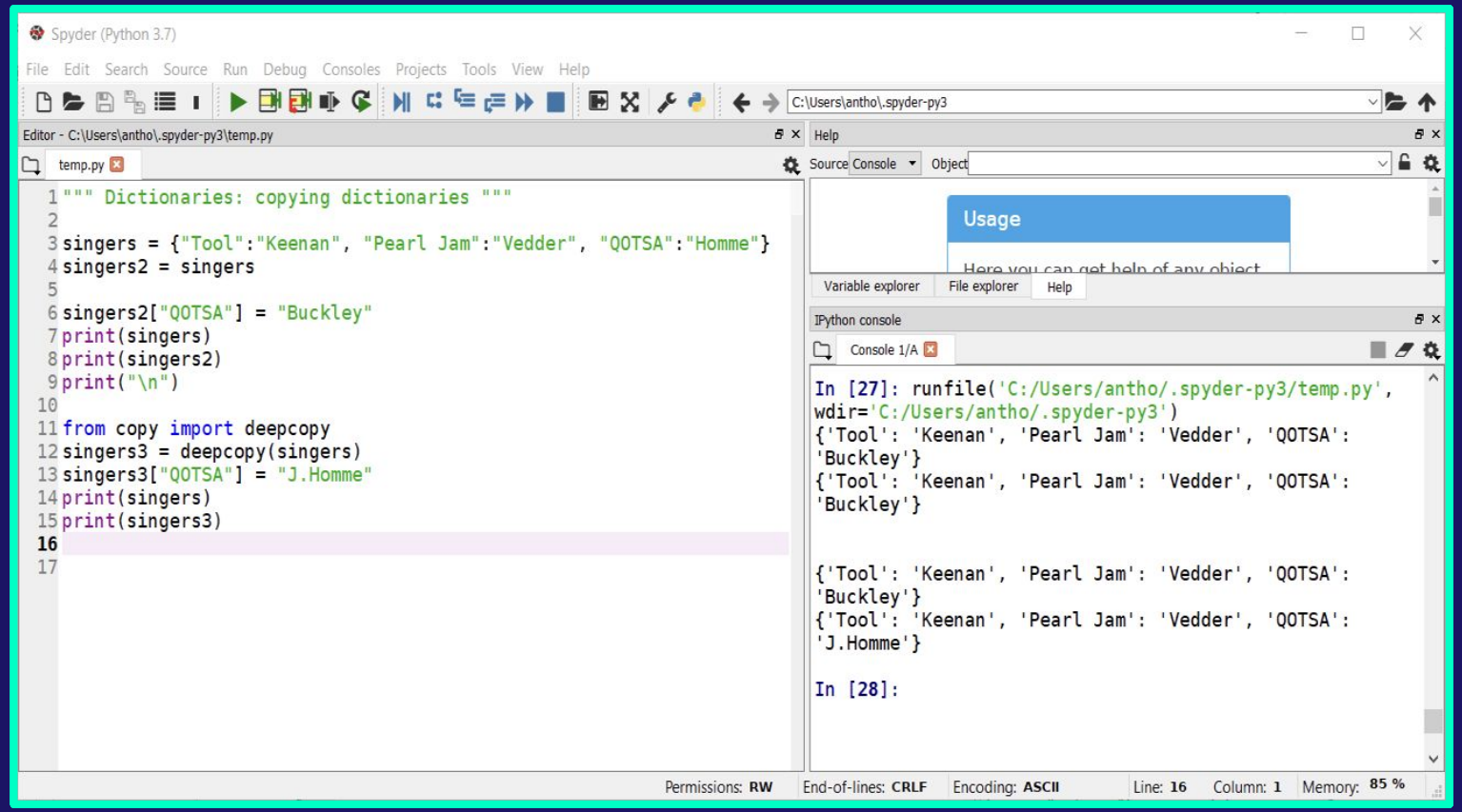
Python Dictionaries (5)



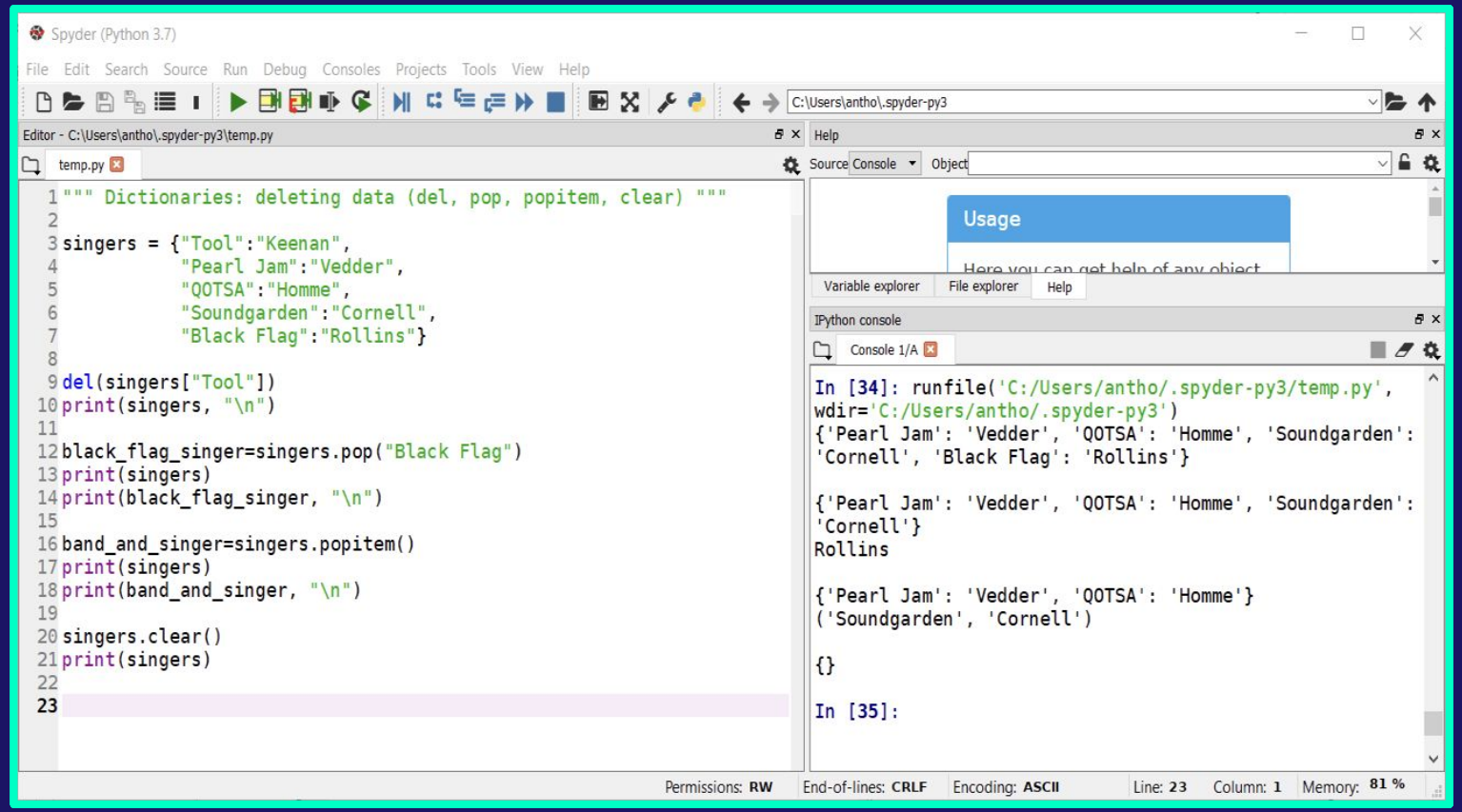
Python Dictionaries (6)



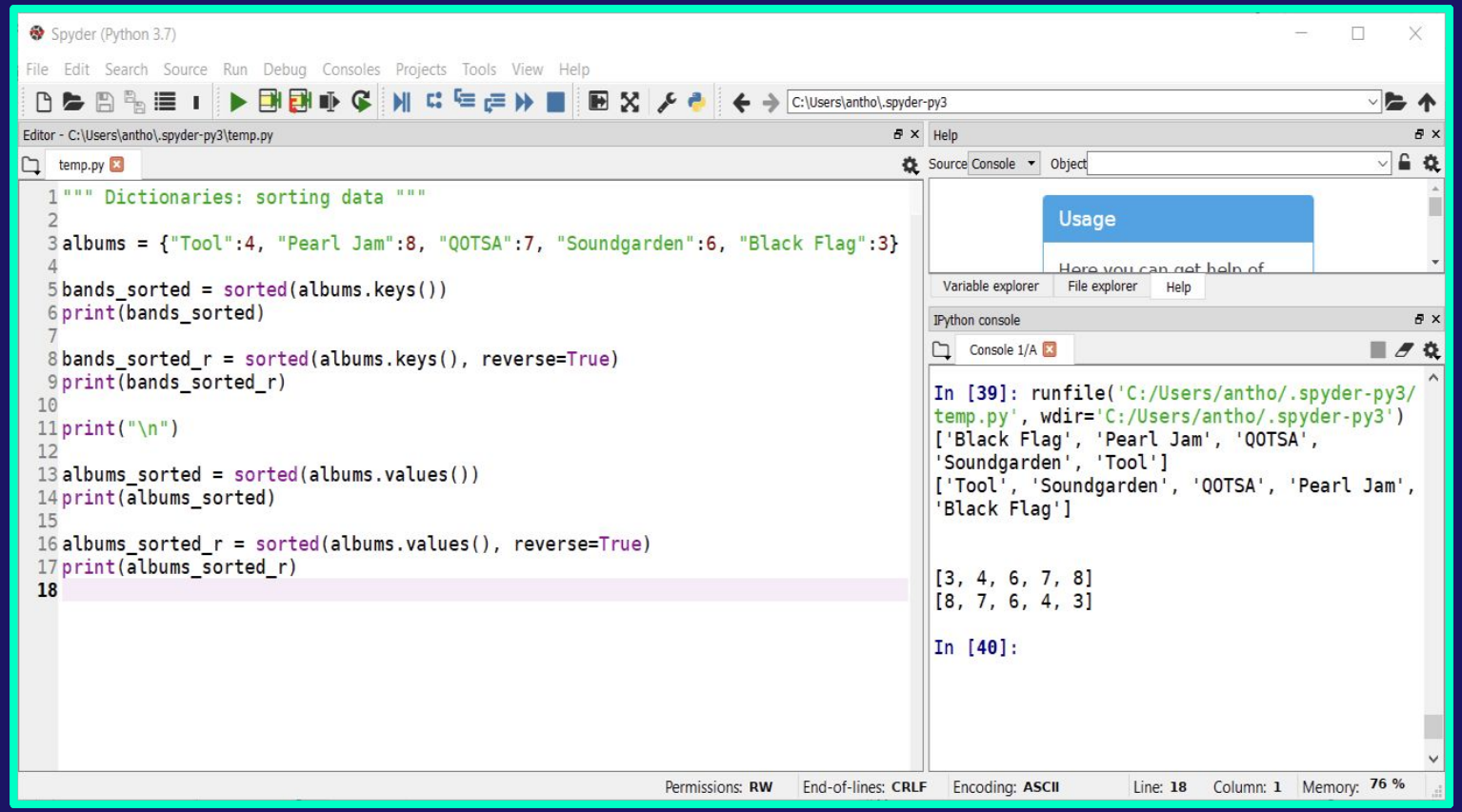
Python Dictionaries (7)



Python Dictionaries (8)



Python Dictionaries (9)



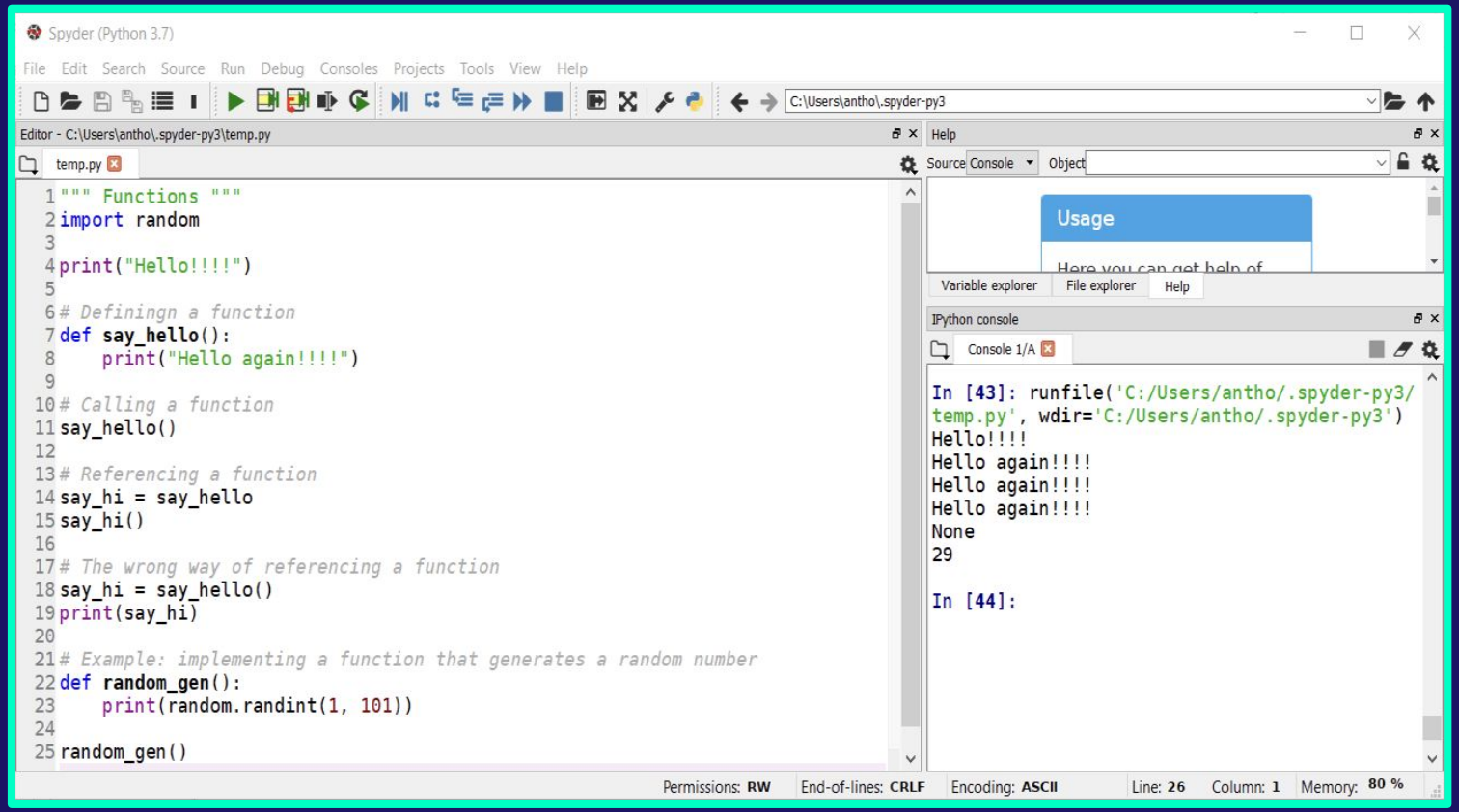
Python Functions (1)

- Functional Programming:
  - Hierarchical design
  - Smaller problems, easier to be solved
  - Development, debugging & maintaining effort & time
  - Less code (reusability)
  - Readability
  - Abstraction

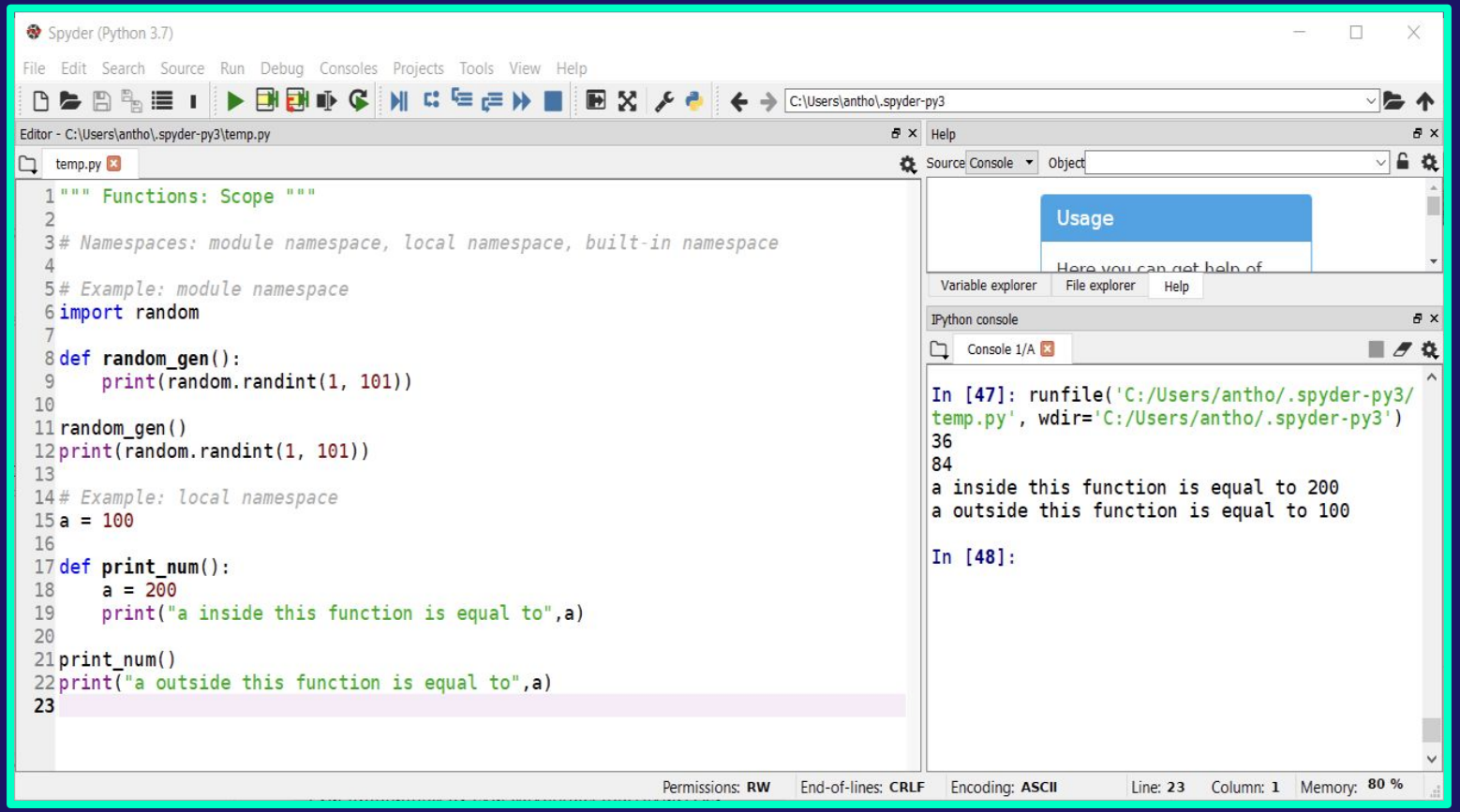
- def function_name(parameters):
  commands
  . . . . .
  commands
  Return_statement

- parameters & return_statement are optional

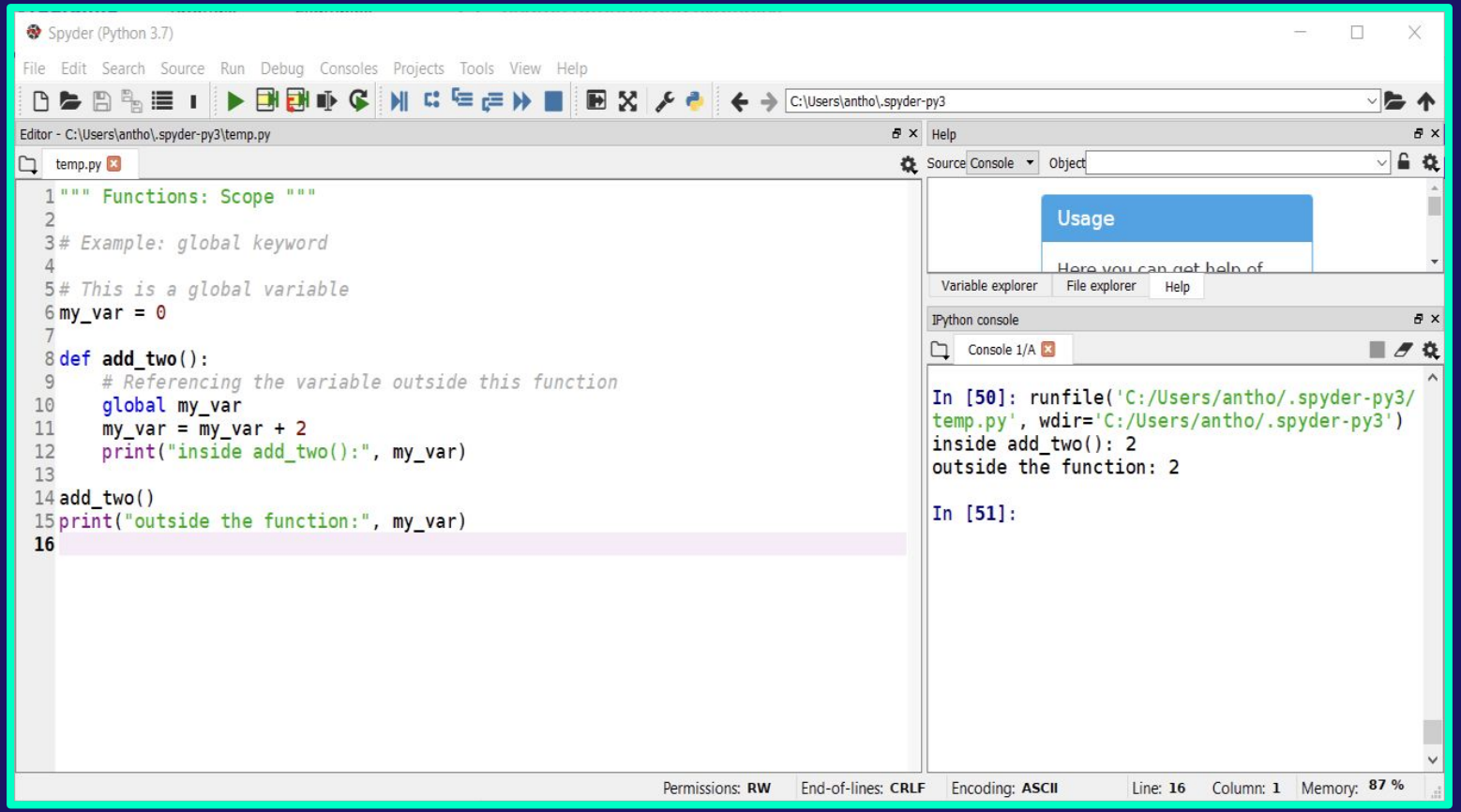
Python Functions (2)



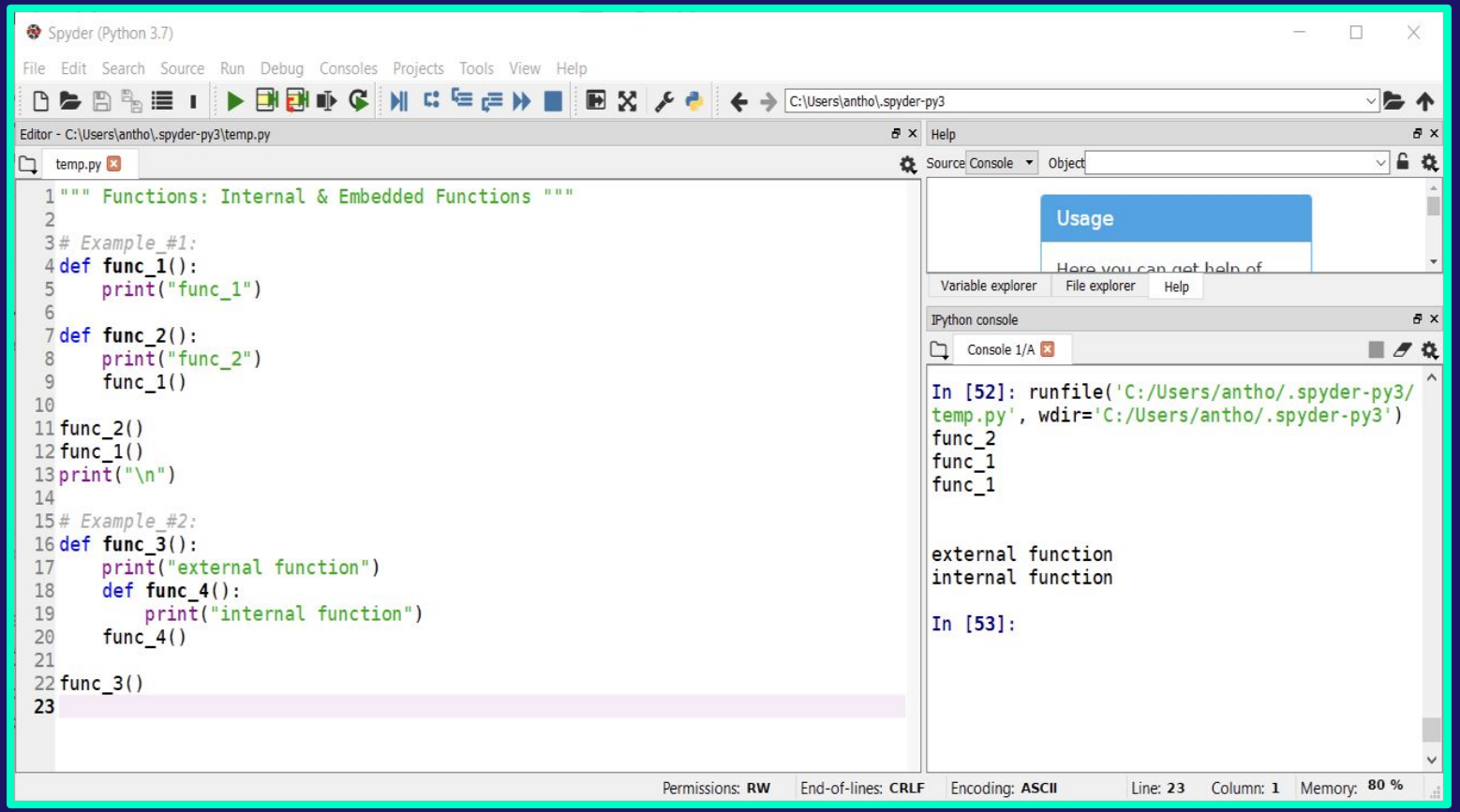
Python Functions (3)



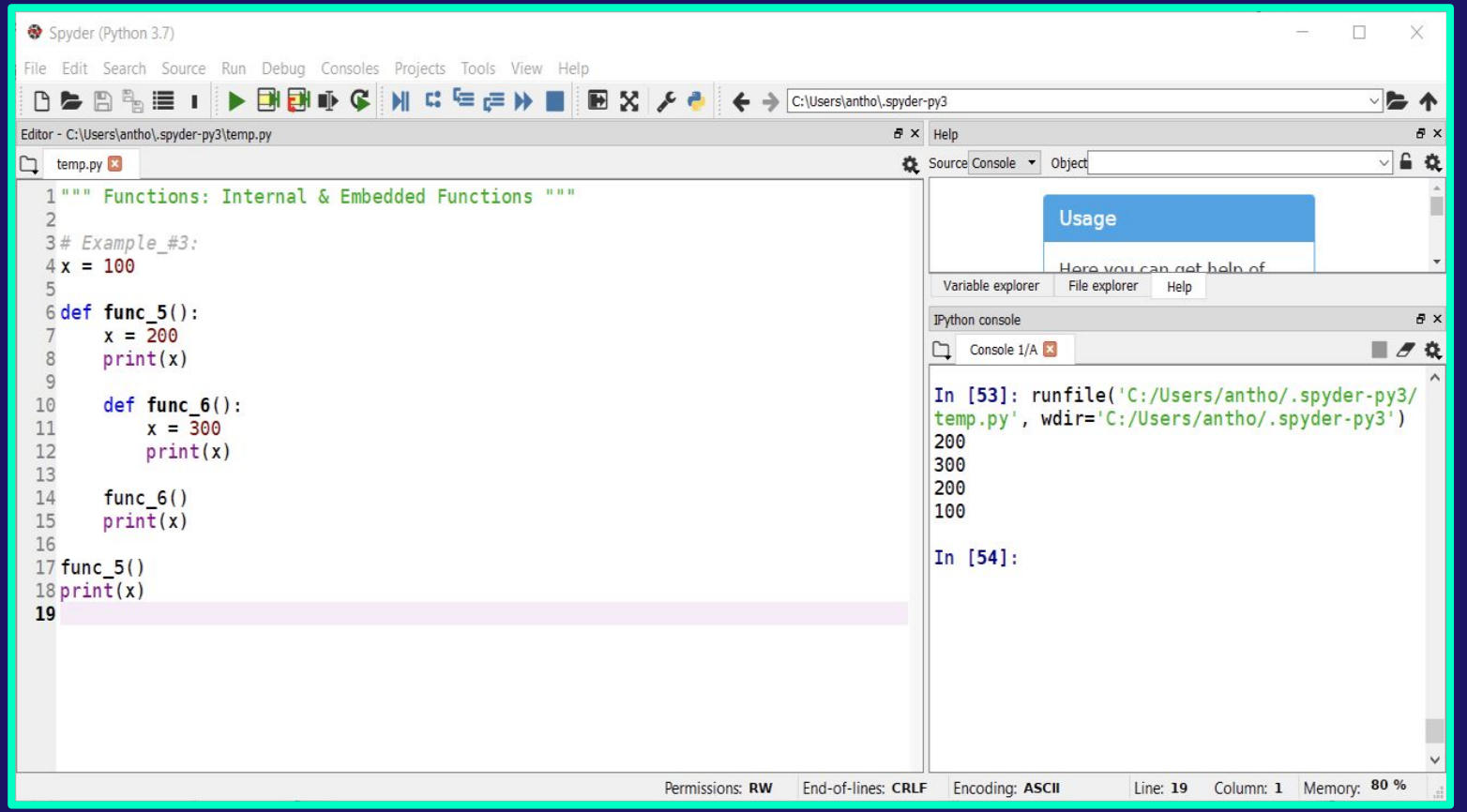
Python Functions (4)



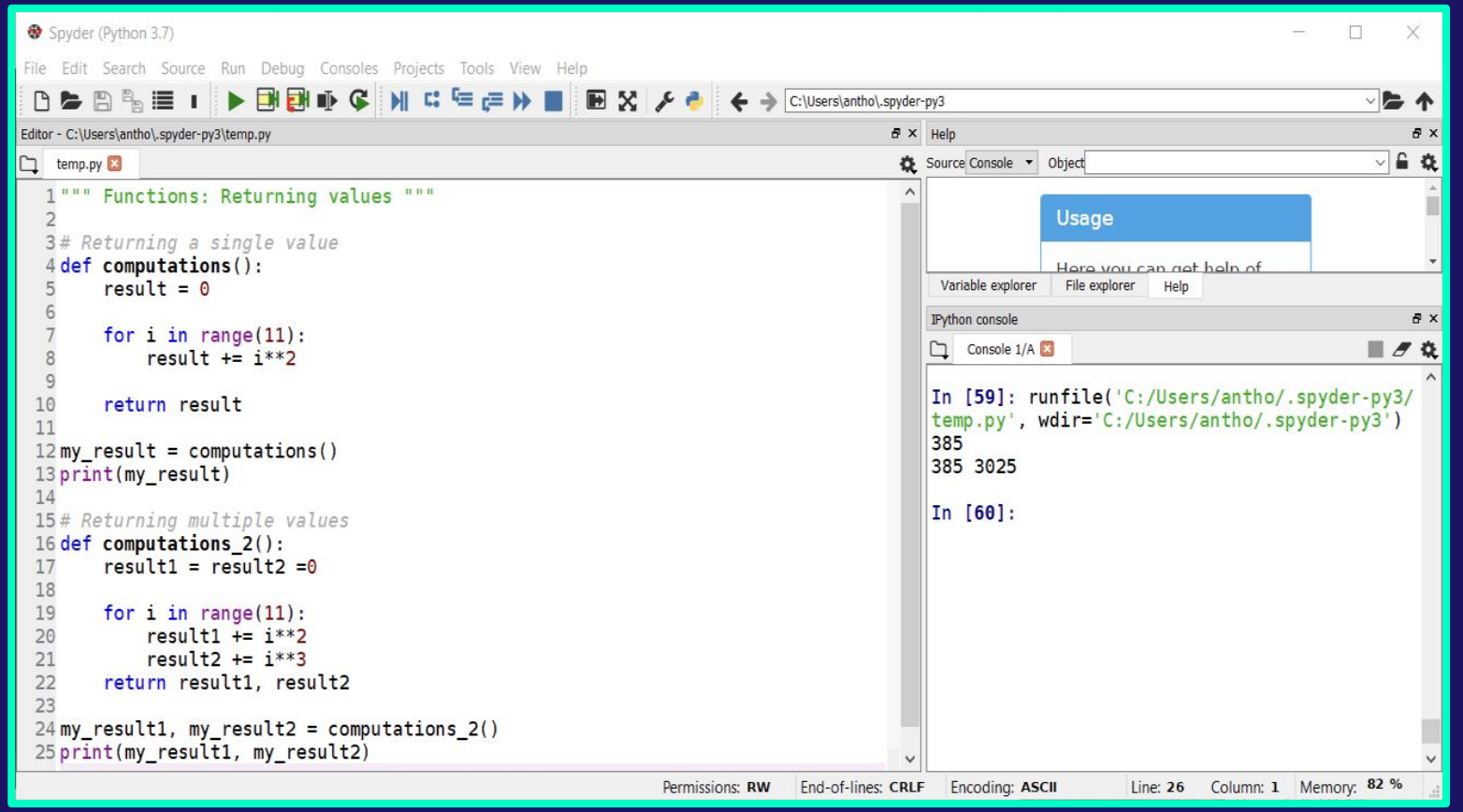
Python Functions (5)



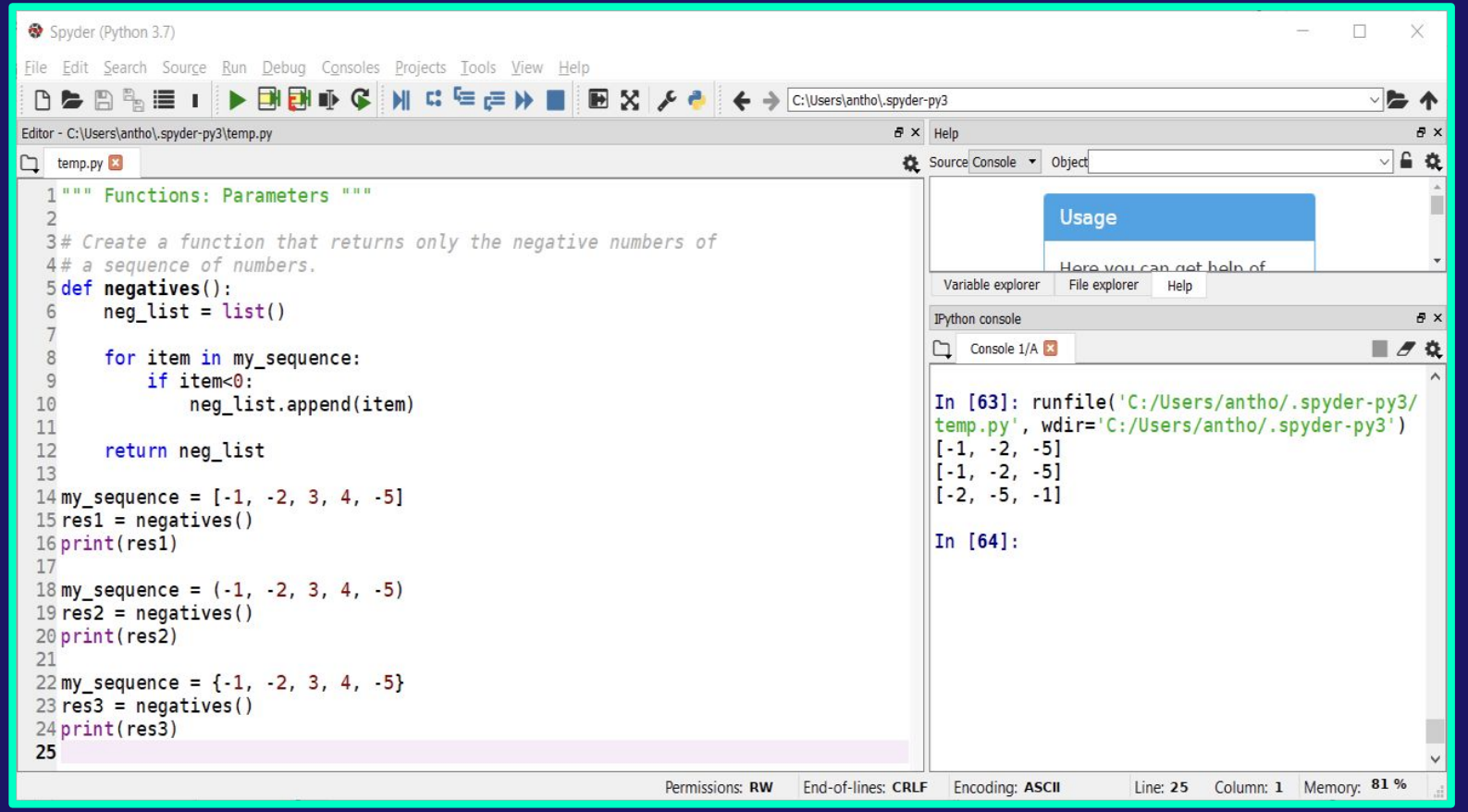
Python Functions (6)



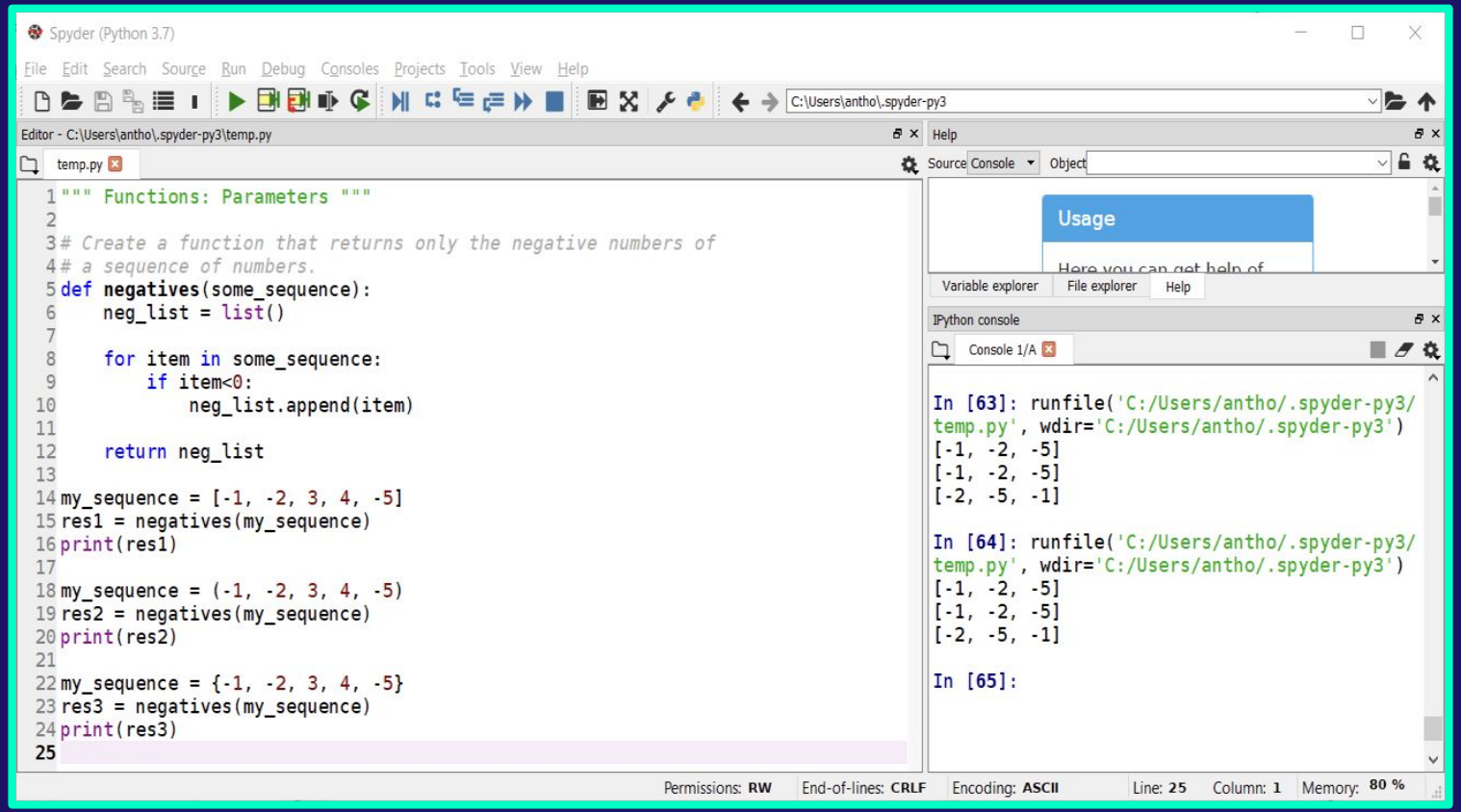
Python Functions (7)



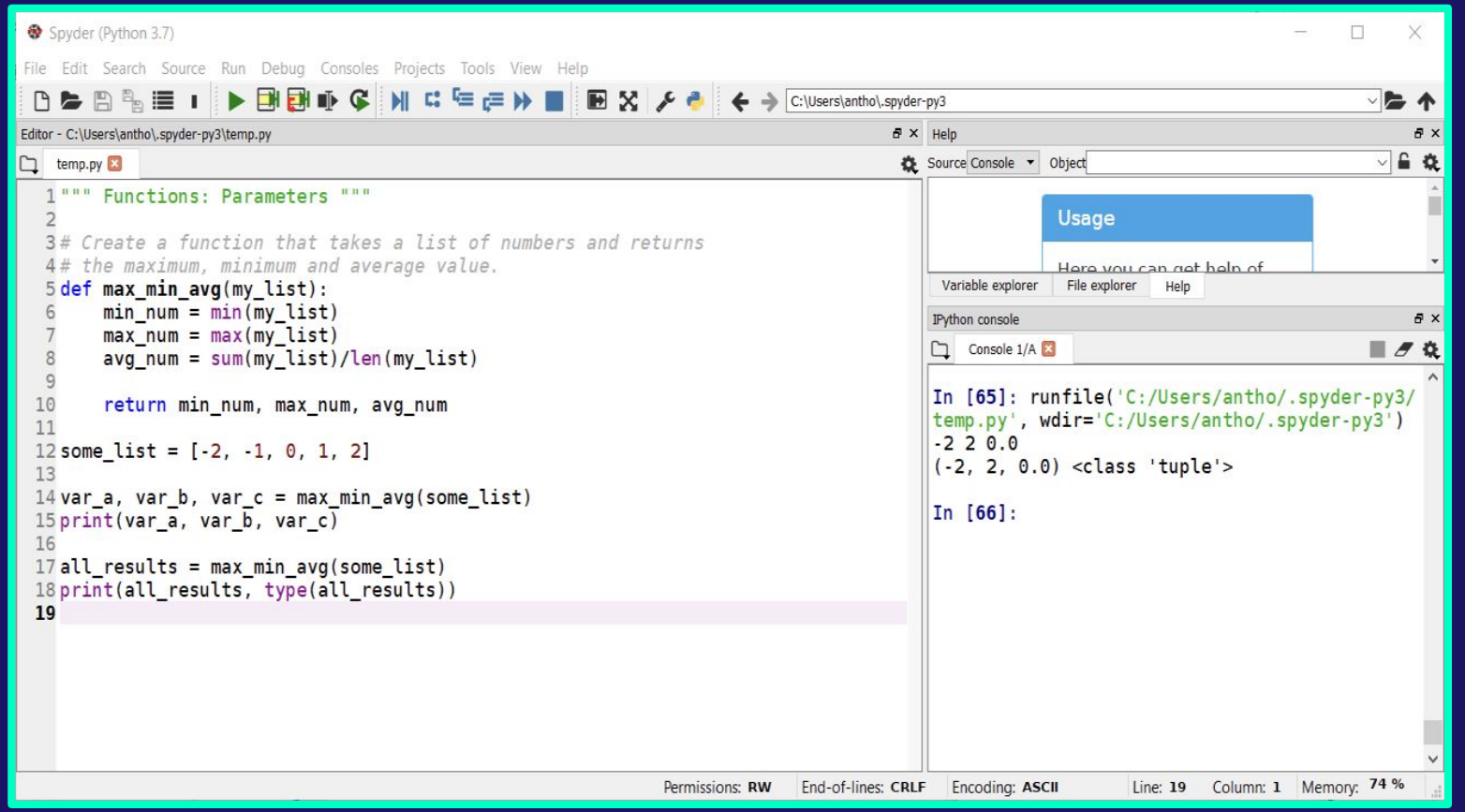
Python Functions (8)



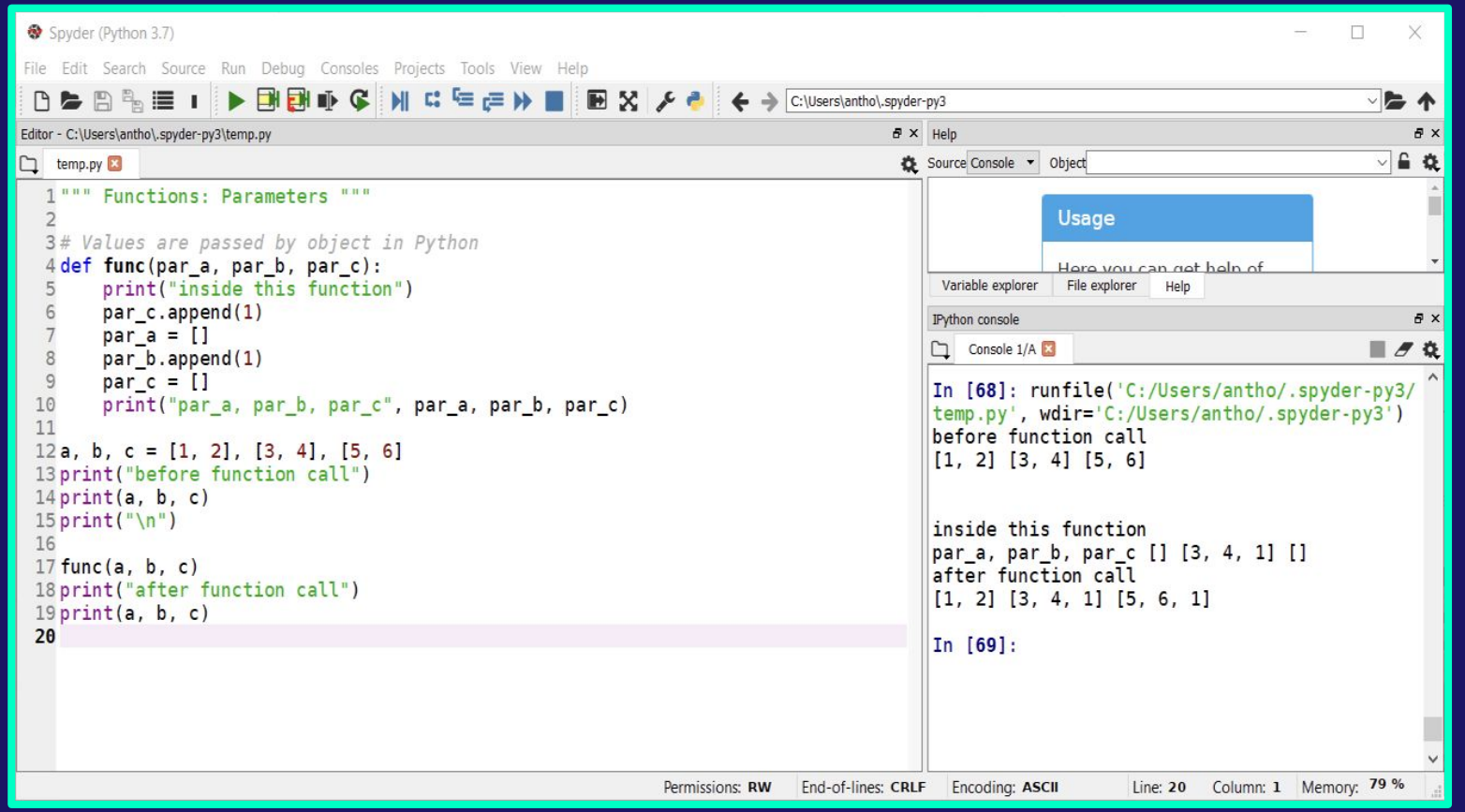
Python Functions (9)



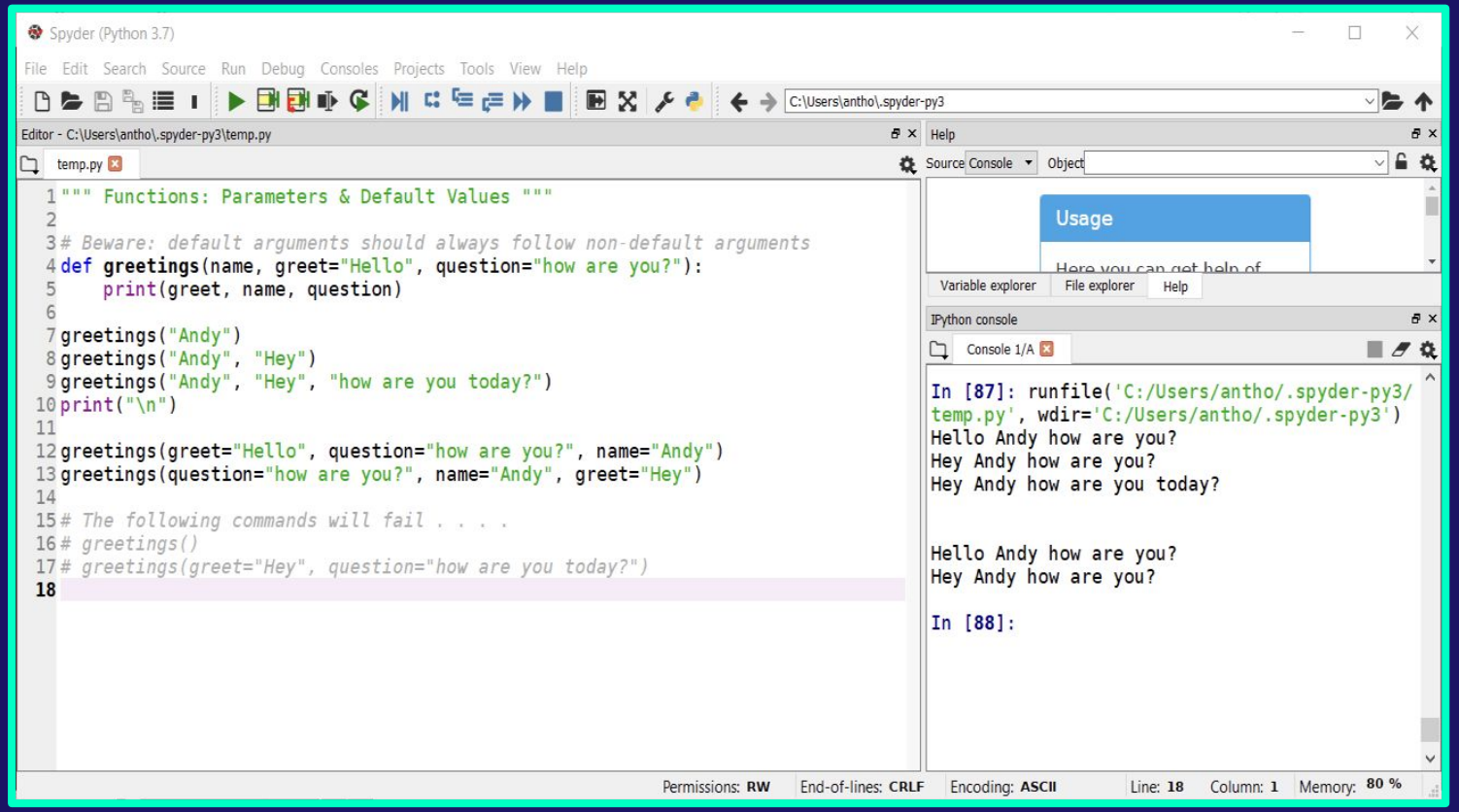
Python Functions (10)



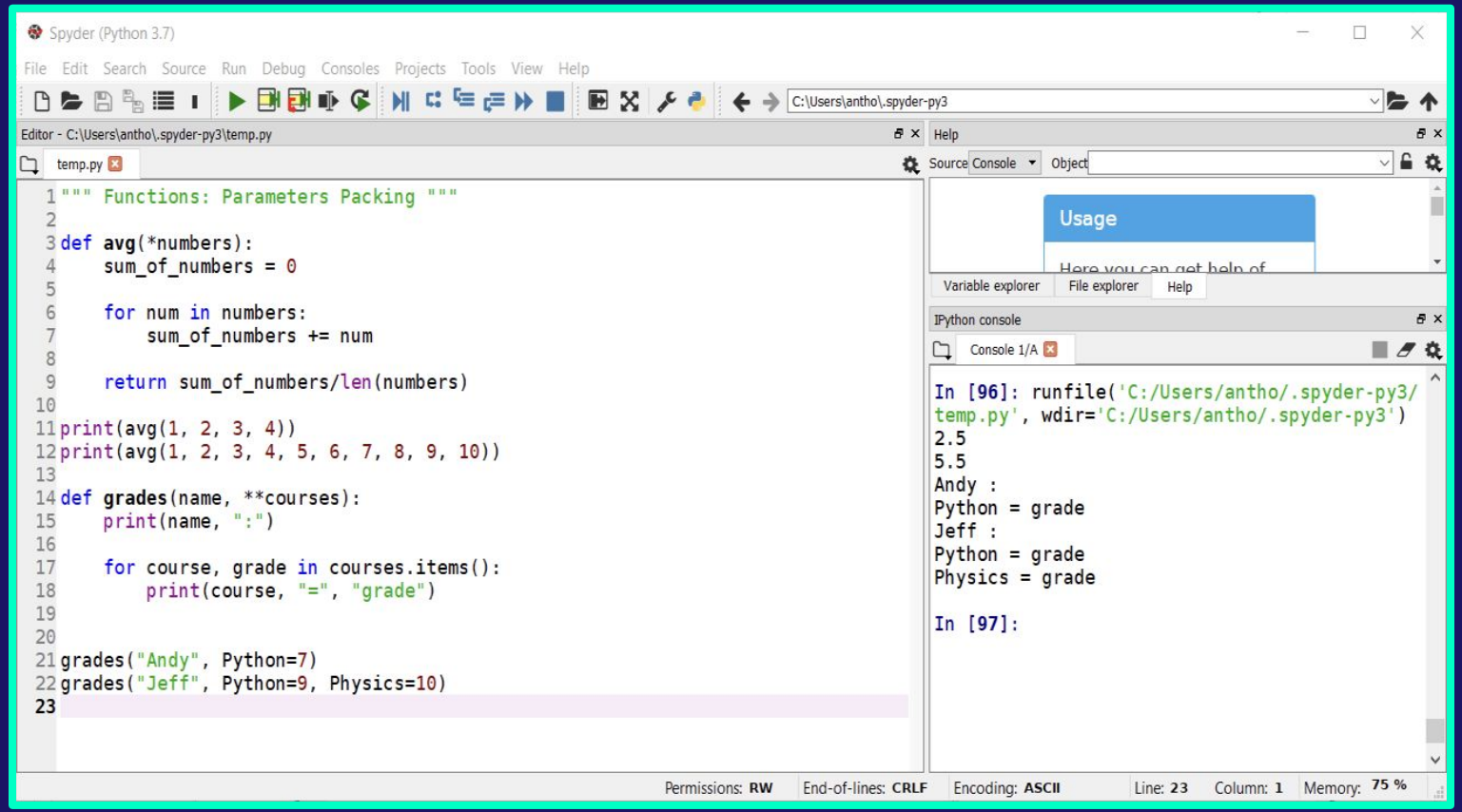
Python Functions (11)



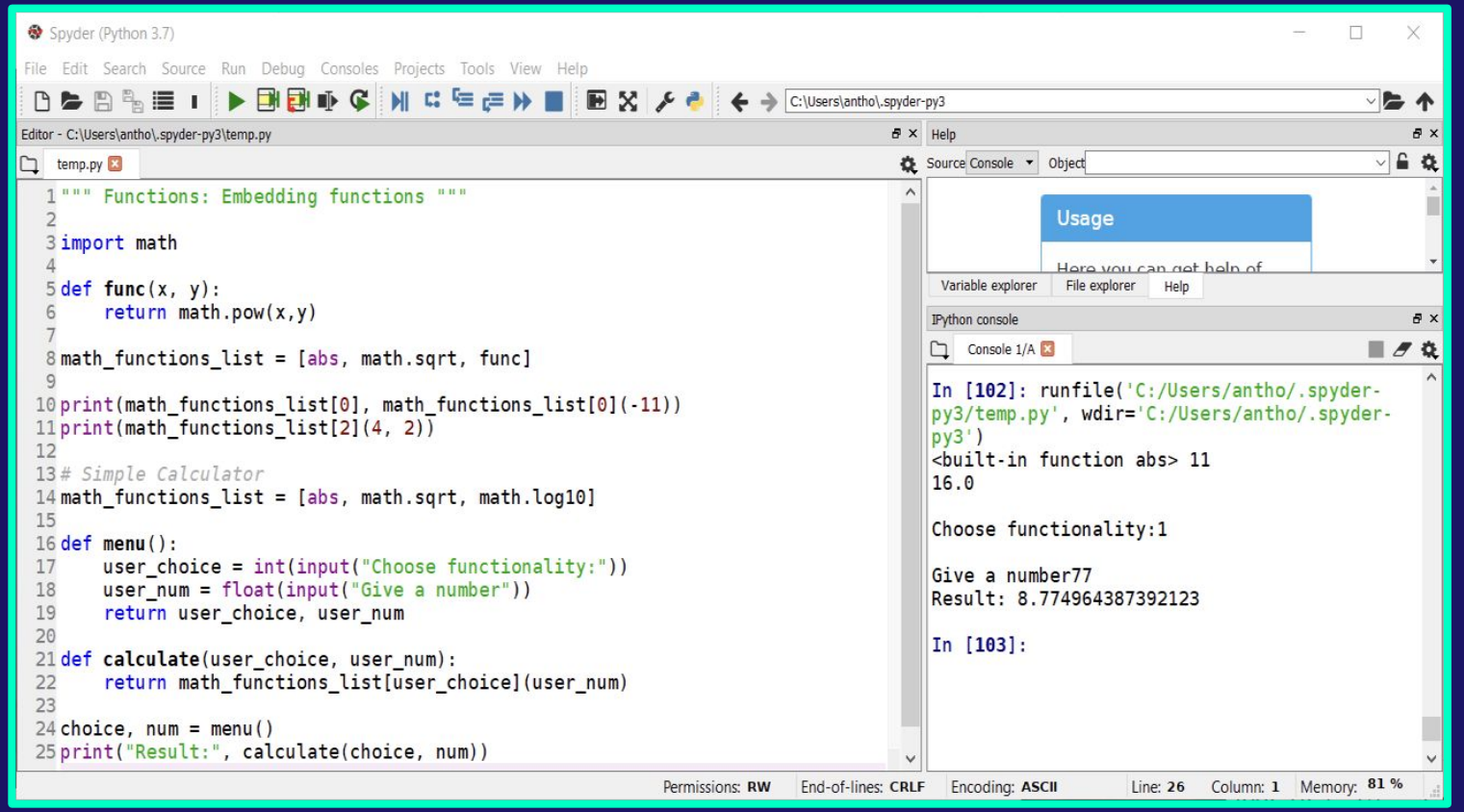
Python Functions (12)



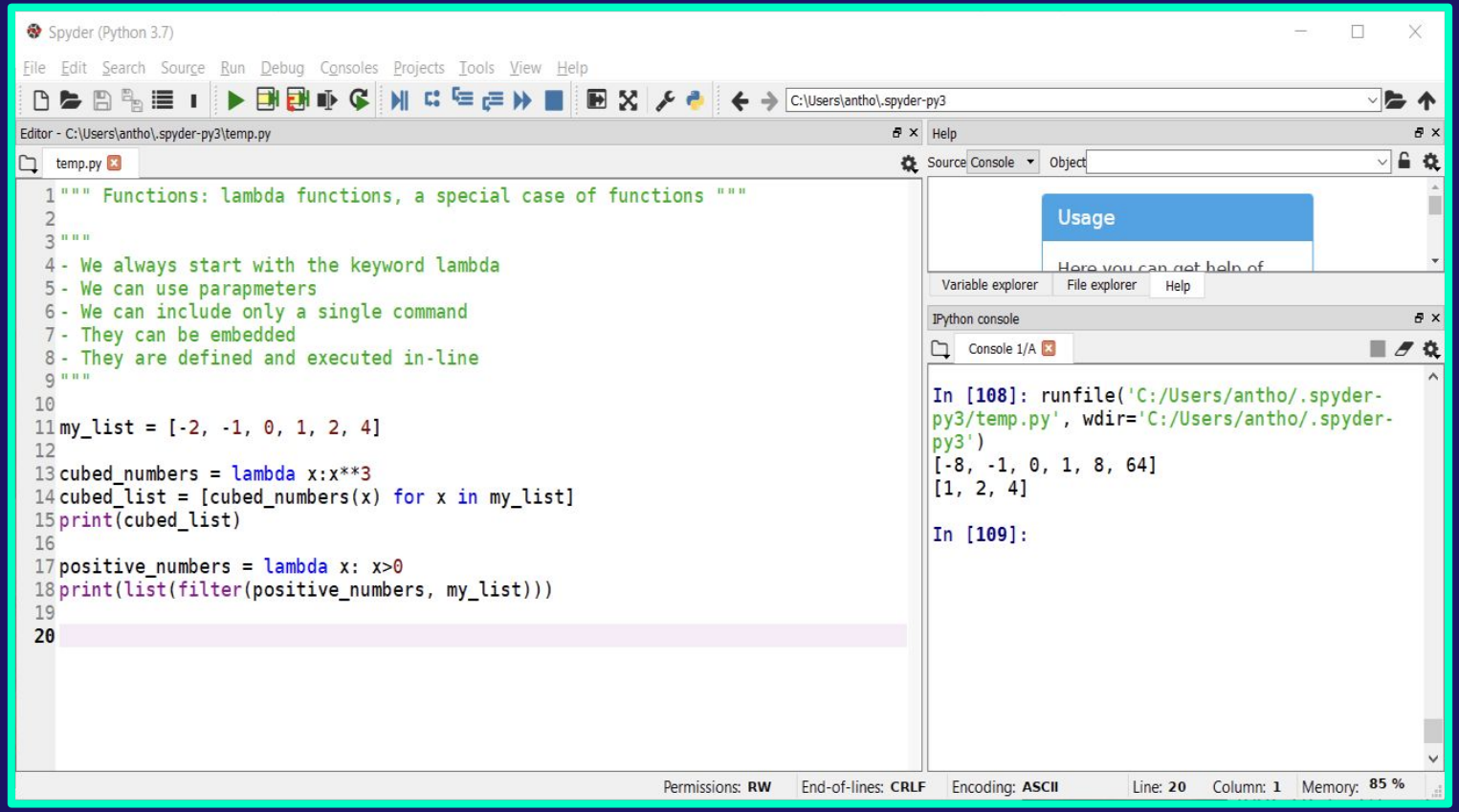
Python Functions (13)



Python Functions (14)



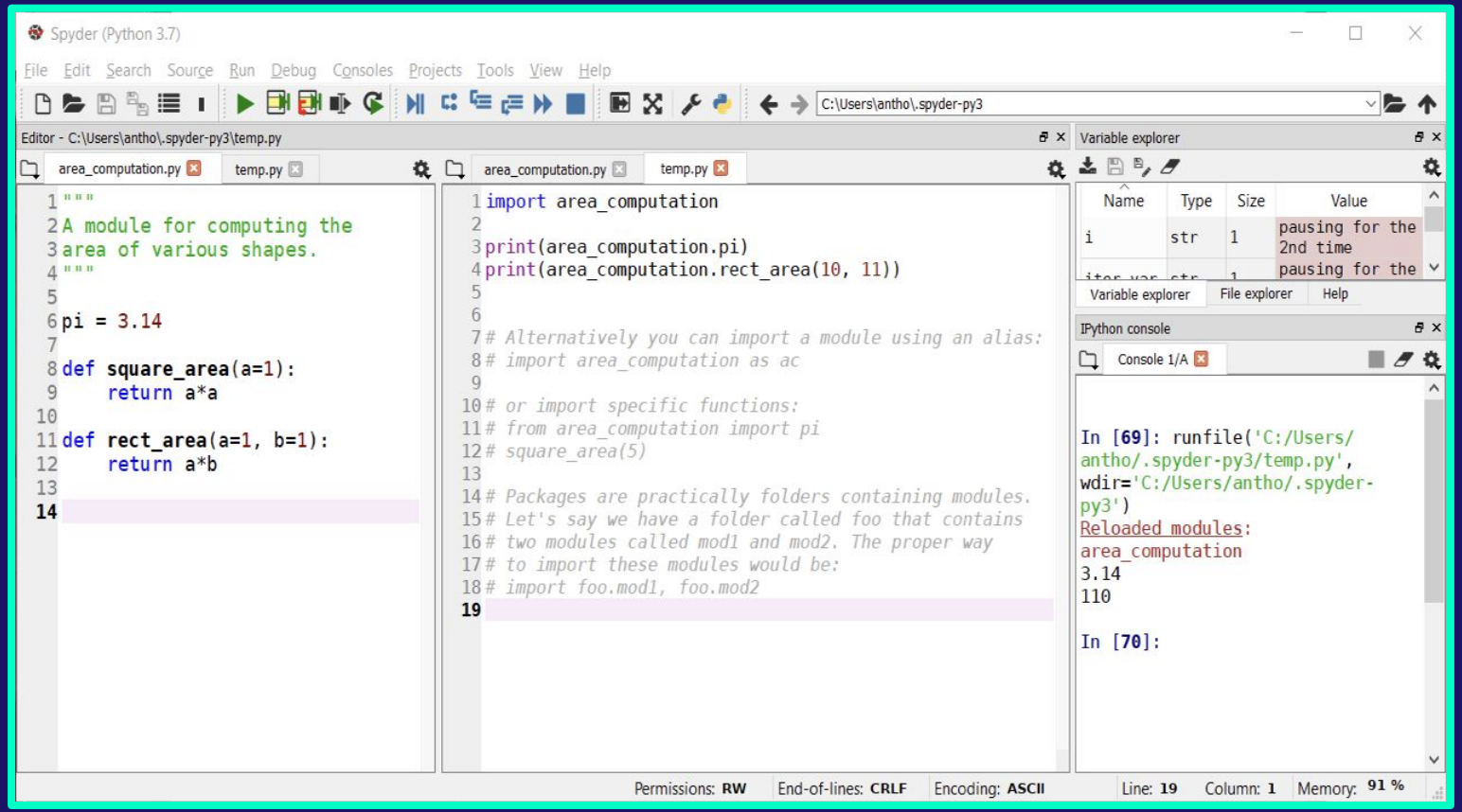
Python Functions (15)



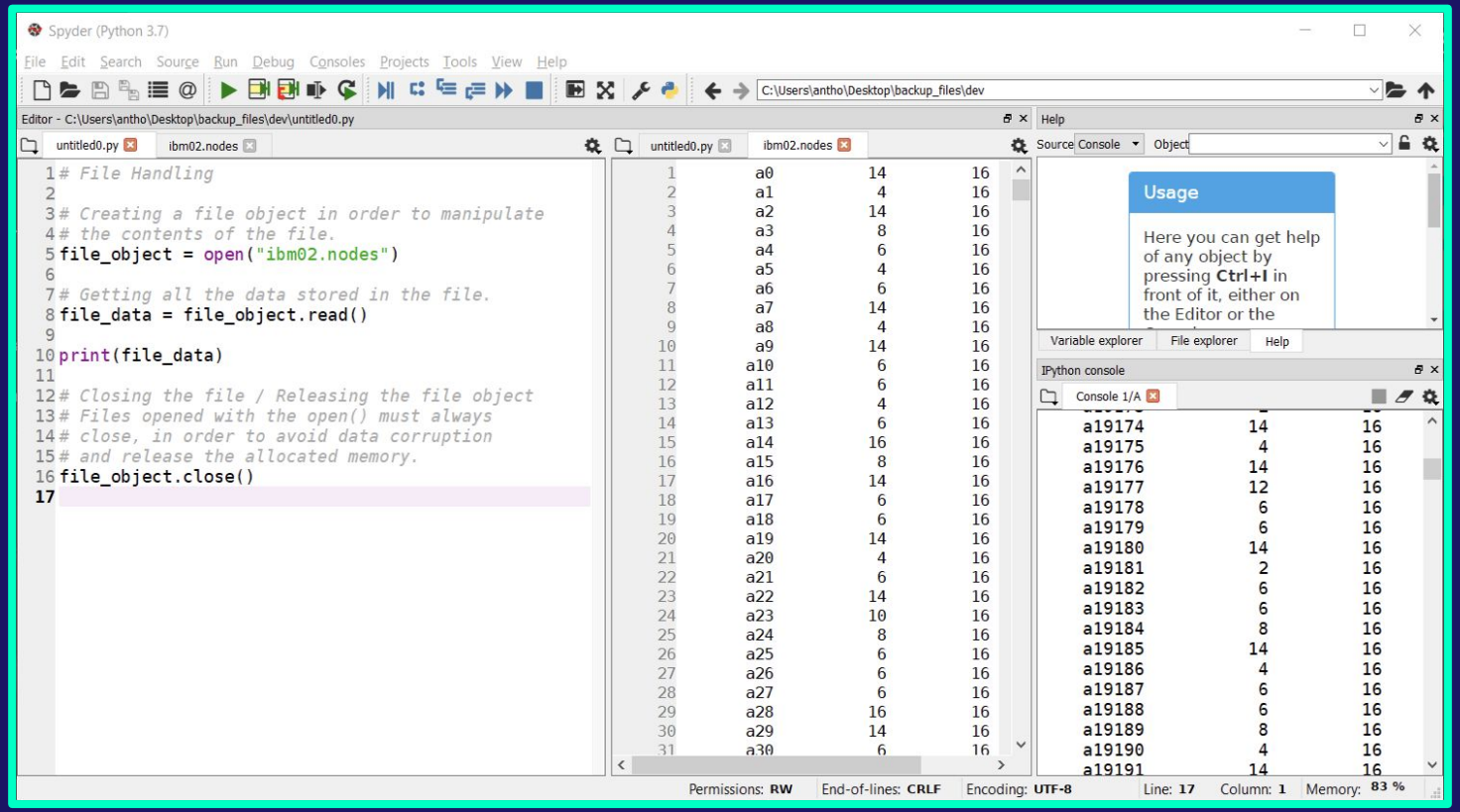
Python Modules (1)

- Python implements the concept of modular programming through modules.
- Module:
  - A Python file (or object file) in our computer that contains code that is available for usage.
    - User-defined modules
    - Built-in modules
    - Code repositories (pip, conda, ...)

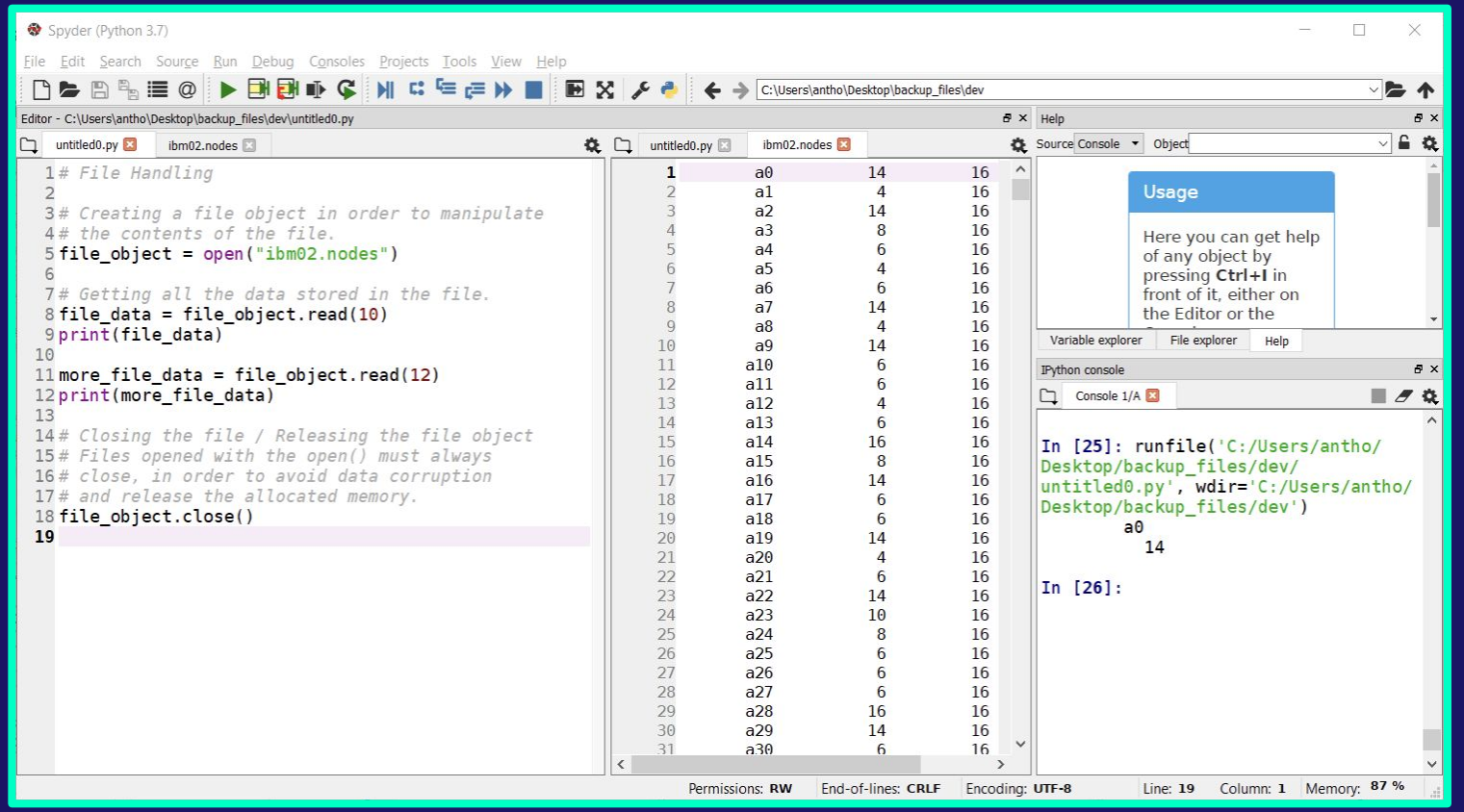
Python Modules (2)



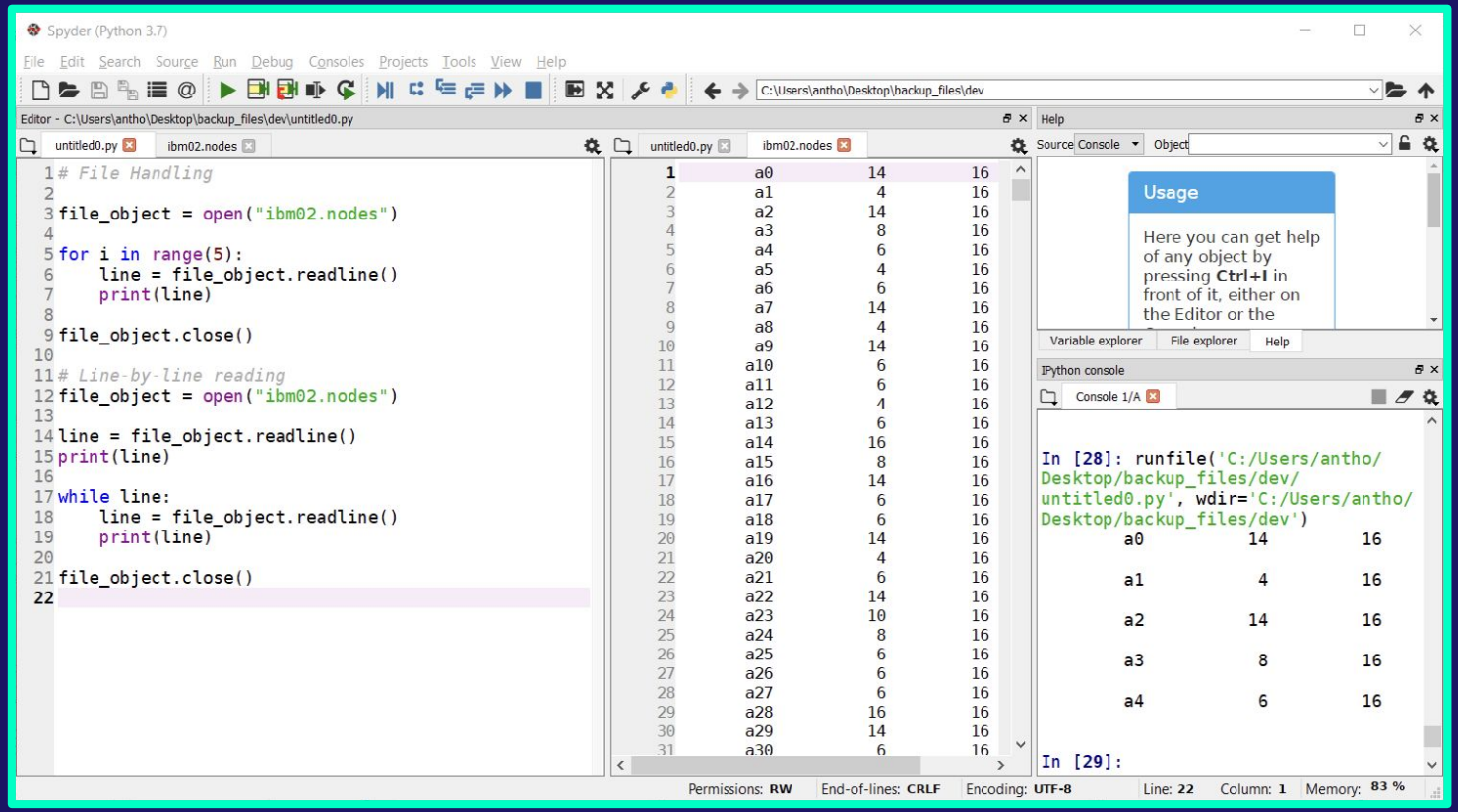
Python File Handling (1)



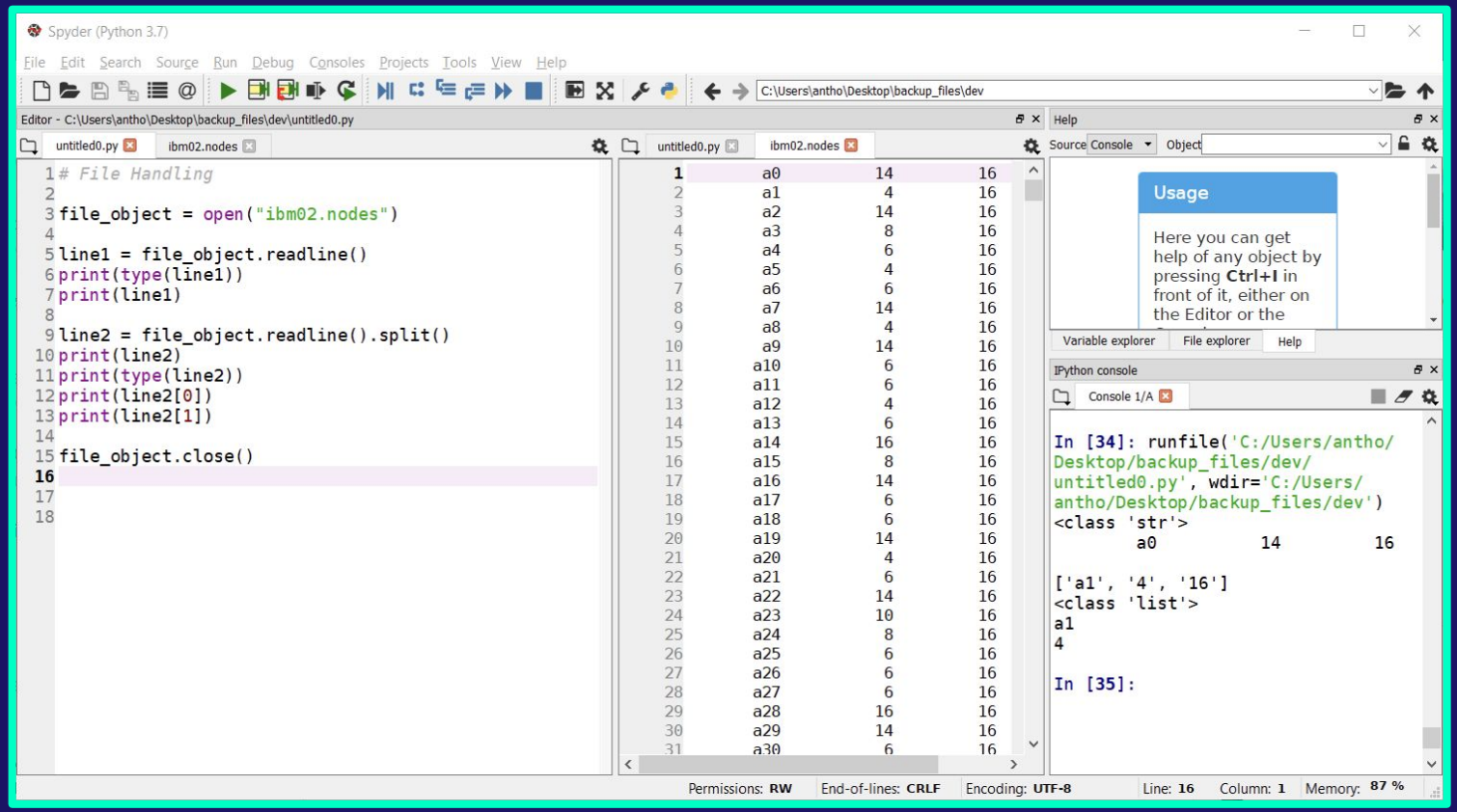
Python File Handling (2)



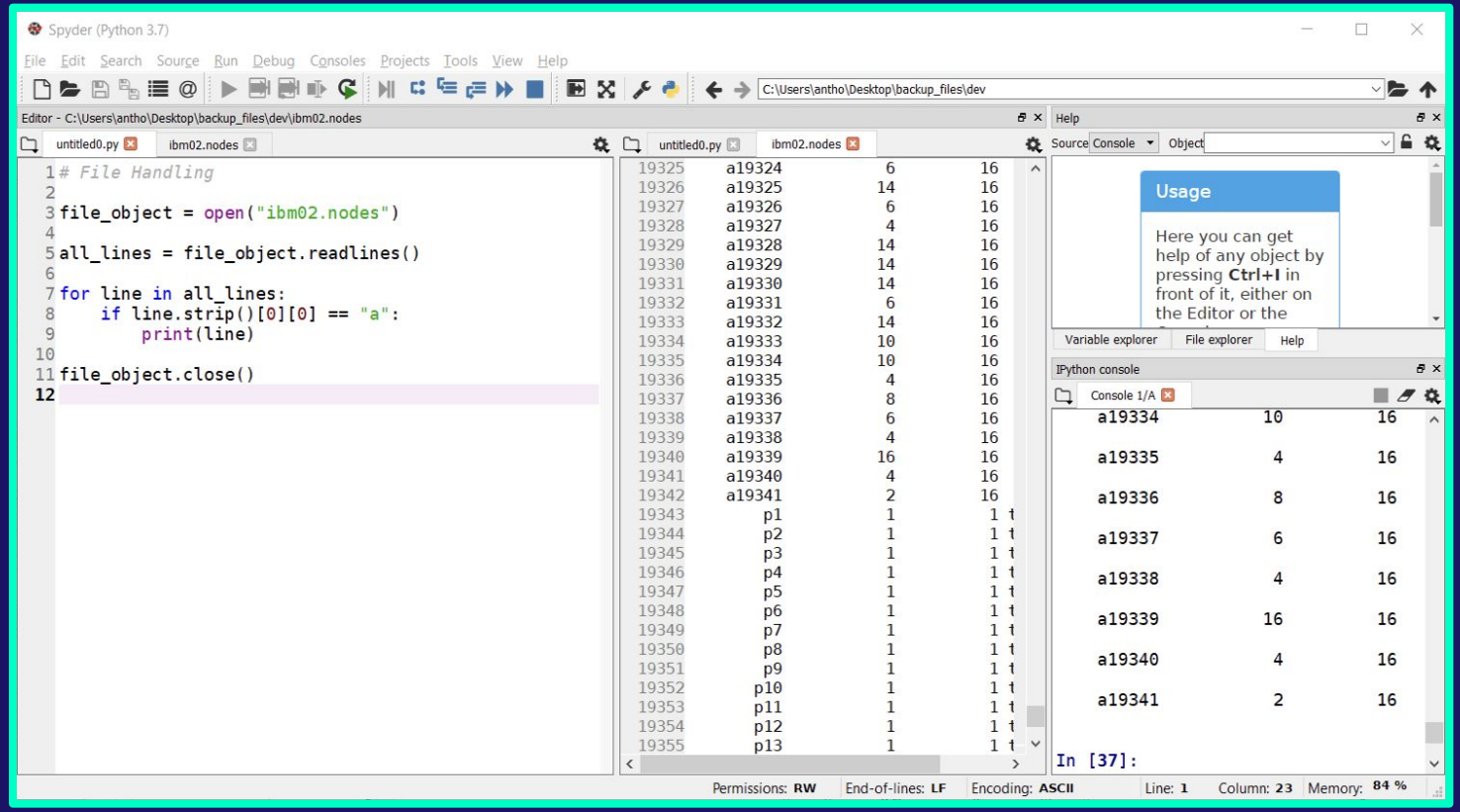
Python File Handling (3)



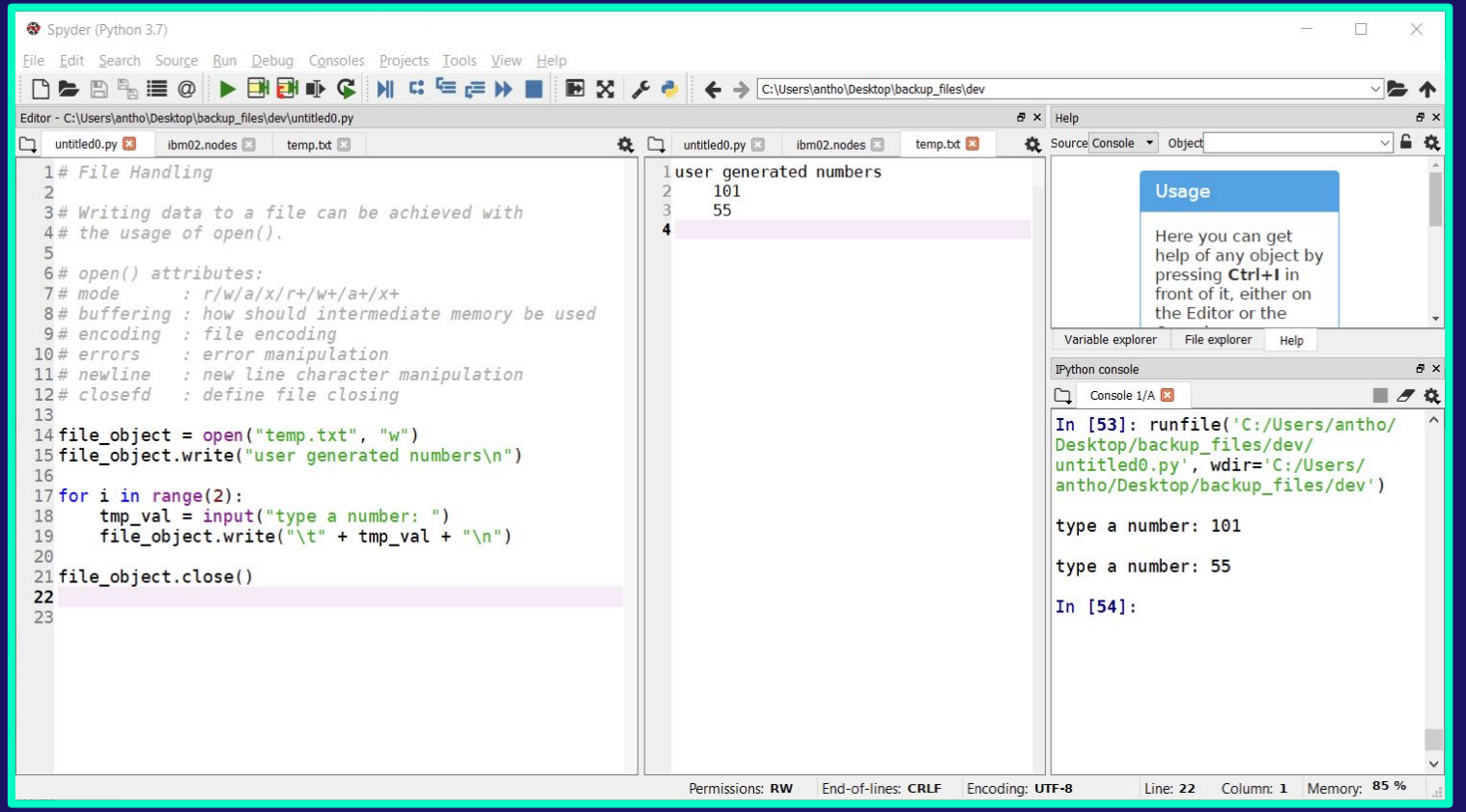
Python File Handling (4)



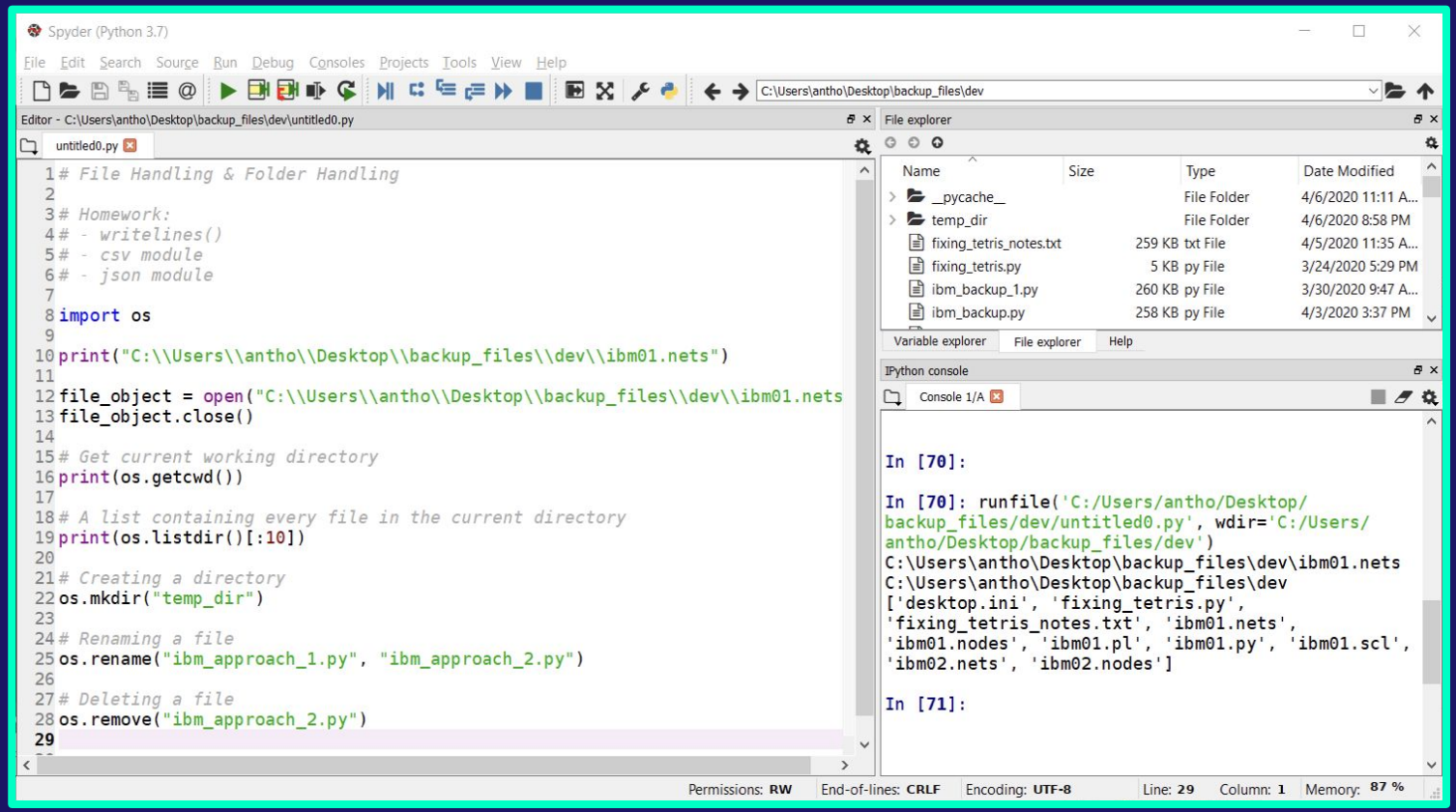
Python File Handling (5)



Python File Handling (6)



Python File Handling (7)



Python Code "Quality" (1)

- Good Code:
  - Does what is supposed to do.
  - Does not contain bugs, errors, logic errors, etc.
  - Readable
  - Maintainable
  - Extensible

- Style Guides:
  - PEP8 (https://pep8.org/) & PEP257 (https://www.python.org/dev/peps/pep-0257/)

- Linters: The "annoying" underlines generated in most IDEs . . . . .

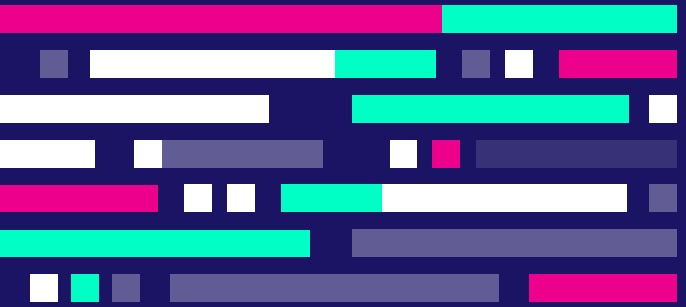
Python Code “Quality” (2)

- Linters (errors, potential errors, code patterns, style):
  - PyLint
  - PyFlakes
  - Bandit
  - MyPy
  - . . . . .
- You should find the appropriate plug-in for the IDE of your choice....

“Laws”, “Rules” & “Principles” related to Software Development/Coding

- Murphy's Law: “If something can go wrong, it will”
- Brook's Law: “Adding manpower to a late software project makes it later”
- Conway’s Law: “Any piece of software reflects the organizational structure that produced it”
- Linus's Law: “Given enough eyeballs, all bugs are shallow”
- Ninety-ninety rule: “The first 90% of the code takes 10% of the time, the remaining 10% takes the other 90% of the time”
- Knuth's optimization principle: “Premature optimization is the root of all evil”

Επαναληπτικές Ασκήσεις



1 - Παρουσιάστε ένα απλό παράδειγμα για κάθε έναν από τους αριθμητικούς τελεστές της γλώσσας.

2 - Δημιουργήστε δύο μεταβλητές με λάθος ονομασία και παρατηρήστε τον τρόπο με τον οποίο παράγονται ειδοποιήσεις στην κονσόλα.

3 - Γράψτε ένα μικρό παράδειγμα χρήσης της εντολής input.

4 - Ποιά θα είναι η τιμή της μεταβλητής a μετά την εκτέλεση των παρακάτω αναθέσεων;

- a = 7-6/4
- a = a/2-a/3-a/4
- a = 1-1/5-5/8-2+11

5 - Γράψτε τις εντολές εκχώρησης ώστε η μεταβλητή a να πάρει τις παρακάτω τιμές:

- Τον μέσο όρο των μεταβλητών b, c, d, e.
- Το υπόλοιπο της διαίρεσης του b με το c, προσαυξημένο κατά d.
- Την απόλυτη τιμή της διαφοράς των c και d.

Επαναληπτικές Ασκήσεις

6 - Γράψτε ένα πρόγραμμα το οποίο δέχεται έναν πενταψήφιο αριθμό ως είσοδο (δεν χρειάζεται να ελεγχθεί το πλήθος των ψηφίων) και επιστρέφει το άθροισμα των ψηφίων του και τον αριθμό ανεστραμμένο.

7 - Γράψτε ένα πρόγραμμα το οποίο υπολογίζει την Ευκλείδεια απόσταση δύο σημείων.

8 - Γράψτε ένα πρόγραμμα το οποίο δέχεται τρεις αριθμούς και επιστρέφει τον μέγιστο, χωρίς τη χρήση της συνάρτησης max.

9 - Γράψτε ένα πρόγραμμα το οποίο παράγει τα 6 νούμερα του λόττο (από 1 έως 49). Ποιό είναι ένα πιθανό πρόβλημα που μπορεί να προκύψει από την εκτέλεση του κώδικα;

10 - Να γραφεί ένα πρόγραμμα το οποίο θα υπολογίζει την ακόλουθη συνάρτηση:

- f(x,y) =
  - $x - y$, $x >= 0, y >= 0$
  - $x^4 - y^3$, $x >= 0, y < 0$
  - $x - |y|$, $x < 0, y < 0$

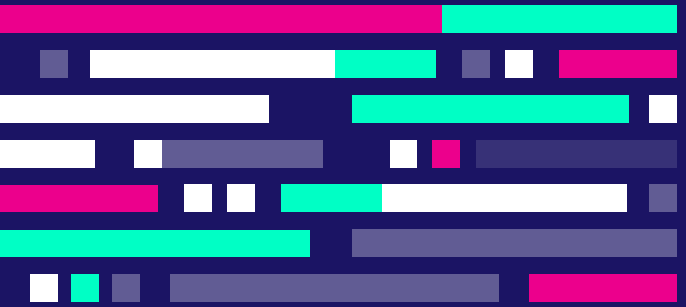
Επαναληπτικές Ασκήσεις

11 - Υλοποιήστε το παιχνίδι πέτρα-ψαλίδι-χαρτί. Οι παίκτες που συμμετέχουν είστε εσείς και ο υπολογιστής (οι επιλογές του υπολογιστή πρέπει να γίνονται με τυχαίο τρόπο). Ο αριθμός των γύρων που θα κρίνουν τον τελικό νικητή πρέπει να δίνεται στην αρχή της εκτέλεσης του προγράμματος από το χρήστη.

12 - Ένας βιομηχανικός ανελκυστήρας μπορεί να δεχθεί μέχρι 10 πακέτα και έως 1000 κιλά βάρος. Να γραφεί κατάλληλο πρόγραμμα το οποίο θα δέχεται κατ'επανάληψη το βάρος κάθε πακέτου και κατά την ολοκλήρωσή του θα εμφανίζει τον συνολικό αριθμό πακέτων και το συνολικό βάρος που έχει μεταφερθεί.

13 - Να γραφεί ένα πρόγραμμα το οποίο θα δέχεται έναν αριθμό n μεγαλύτερο του 10 στην είσοδό του και θα εμφανίζει όλους τους αριθμούς από το 2 μέχρι το n που είναι πρώτοι.

Επαναληπτικές Ασκήσεις



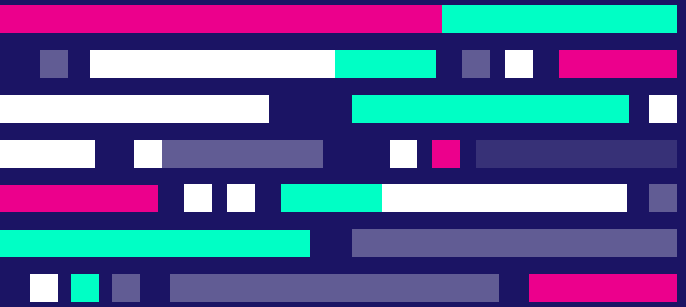
14 - Να γραφεί ένα πρόγραμμα το οποίο θα ζητάει το password του χρήστη προτού προχωρήσει σε κάποια άλλη λειτουργία. Σε περίπτωση τριών αποτυχημένων προσπαθειών το πρόγραμμα θα “κλειδώνει” για ένα λεπτό. (tip: χρησιμοποιήστε την συνάρτηση sleep του module time).

15 - Να γραφεί ένα πρόγραμμα το οποίο θα παράγει μια λίστα που θα περιλαμβάνει τις τετραγωνικές ρίζες των άρτιων αριθμών μιας λίστας.

16 - Να γραφεί ένα πρόγραμμα το οποίο θα διαβάζει μια λίστα με αριθμούς από το 1 έως το 100 (επιτρέπονται τα διπλότυπα) και θα επιστρέφει μια νέα λίστα με τη συχνότητα εμφάνισης των αριθμών.

17 - Να γραφεί ένα πρόγραμμα το οποίο θα εμφανίζει την συχνότητα εμφάνισης του κάθε χαρακτήρα ενός κειμένου που θα δίνεται από τον χρήστη.

Επαναληπτικές Ασκήσεις



18 - Διαβάστε και υλοποιήστε ένα παράδειγμα χρήσης του αλγορίθμου κρυπτογράφησης του Καίσαρα.

19 - Να γραφεί ένα πρόγραμμα το οποίο θα αφαιρεί από ένα κείμενο όλα τα tabs και τα κενά και θα τα αντικαθιστά με underscores.

20 - Να γραφεί ένα πρόγραμμα το οποίο θα επιστρέφει τον αριθμό συμφώνων και φωνηέντων ενός κειμένου.

21 - Να γραφεί ένα πρόγραμμα το οποίο θα δέχεται έναν κωδικό από τον χρήστη και θα κρίνει την δυνατότητα χρήσης του βάσει των παρακάτω προϋποθέσεων:

- πρέπει να αποτελείται από 8 έως 11 χαρακτήρες
- πρέπει να περιλαμβάνει
  - τουλάχιστον ένα κεφαλαίο γράμμα
  - έναν αριθμό και ένα special character

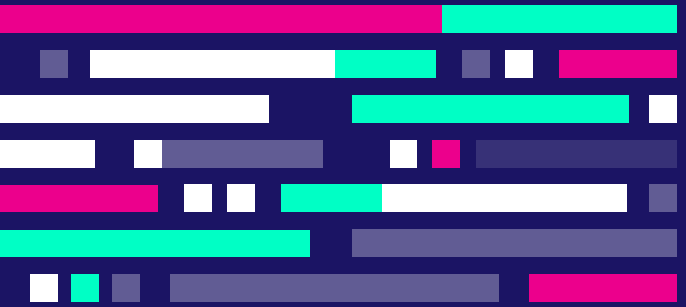
Επαναληπτικές Ασκήσεις

22 - Να γραφεί ένα πρόγραμμα το οποίο θα διαβάζει ένα κείμενο που δίνεται από τον χρήστη και θα επιστρέφει όλους τους διαφορετικούς χαρακτήρες που εμφανίζονται μέσα σε αυτό.

23 - Να γραφεί ένα πρόγραμμα το οποίο θα δέχεται ένα κείμενο από τον χρήστη και θα δημιουργεί ένα λεξικό με κλειδιά τα γράμματα του κειμένου και τιμές τον αριθμό εμφάνισής τους.

24 - Να γραφεί ένα πρόγραμμα το οποίο δημιουργεί ένα λεξικό βάσει των πληροφοριών που εισάγει ο χρήστης και αφορούν κάποιο προϊόν (κωδικός προϊόντος και τιμή). Η διαδικασία θα ολοκληρώνεται όταν ο χρήστης δώσει αρνητική τιμή για κάποιο προϊόν. Μετά το πέρας της παραπάνω διαδικασίας θα εμφανίζεται στην οθόνη το προϊόν με τη μεγαλύτερη τιμή, το προϊόν με τη μικρότερη τιμή και η μέση τιμή προϊόντος.

Επαναληπτικές Ασκήσεις



25 - Να γραφεί μια συνάρτηση που θα παράγει τυχαίους αριθμούς από το 1 έως και το 100, φροντίζοντας να μην επαναλαμβάνεται ο ίδιος αριθμός στην ίδια κλήρωση.

26 - Να γραφεί μια συνάρτηση που θα δέχεται ένα αλφαριθμητικό και θα επιστρέφει ένα λεξικό με κλειδιά τους χαρακτήρες του κειμένου και τιμές το πλήθος εμφανίσεων του κάθε χαρακτήρα.

27 - Να γραφεί ο αλγόριθμος σειριακής αναζήτησης με την μορφή συνάρτησης.

28 - Να γραφεί μια συνάρτηση που θα ελέγχει κατά πόσο ένας αριθμός είναι τέλειος.