TIL6022
TIL Programming | Python

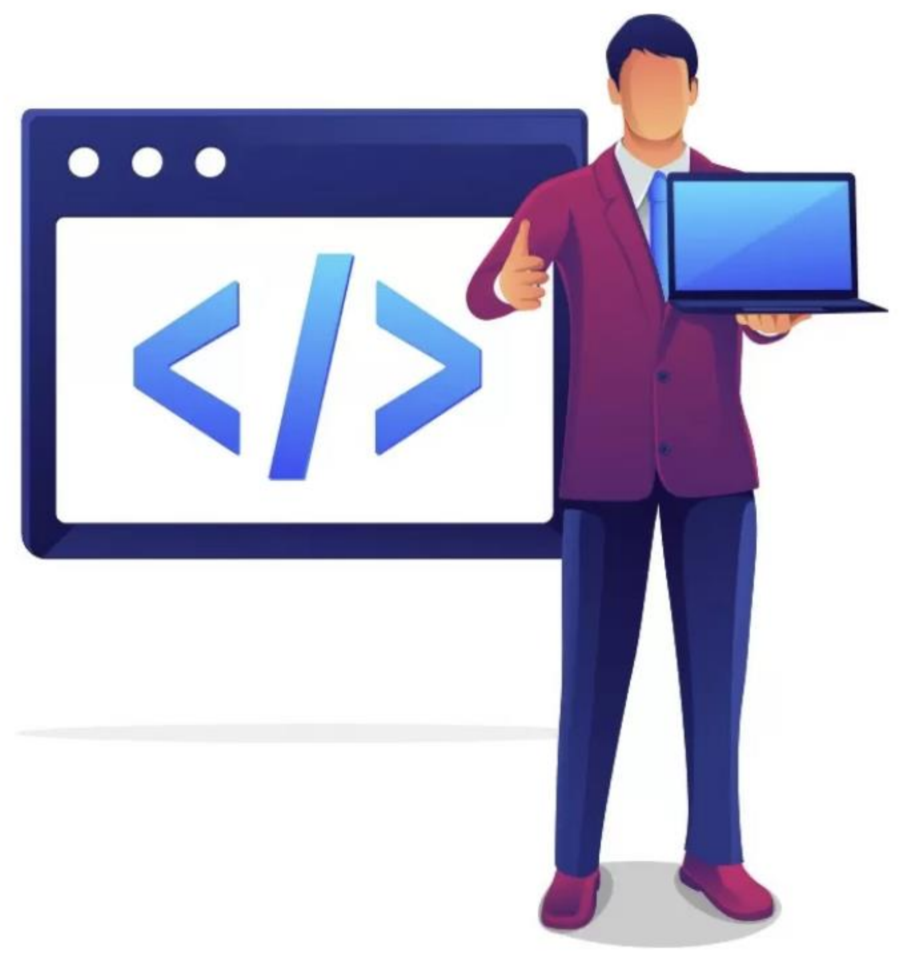
Python Fundamentals II

```
>>> arr = [1, 2, 3, 4, 5]
>>> list(map(lambda x: x * 2, filter(lambda x: x % 2 == 0, arr)))
"""
[4, 8]
"""
```

Dr. Ali Nadi
9 September 2024

Recap Fundamental I

- Data Structures in Python
  - Primitive types
  - Built-in data structures (e.g. Sets)
  - Classes
- Python built-in operations
  - Keywords
  - Operators
- OS Library



What do you Learn?

1. **Control Flow**
   - Conditional statements
   - Loops

2. **Function**
   - Regular functions
   - Anonymous functions
   - Generators functions
   - recursive functions

3. **Exception Handling**
   - Try-exception

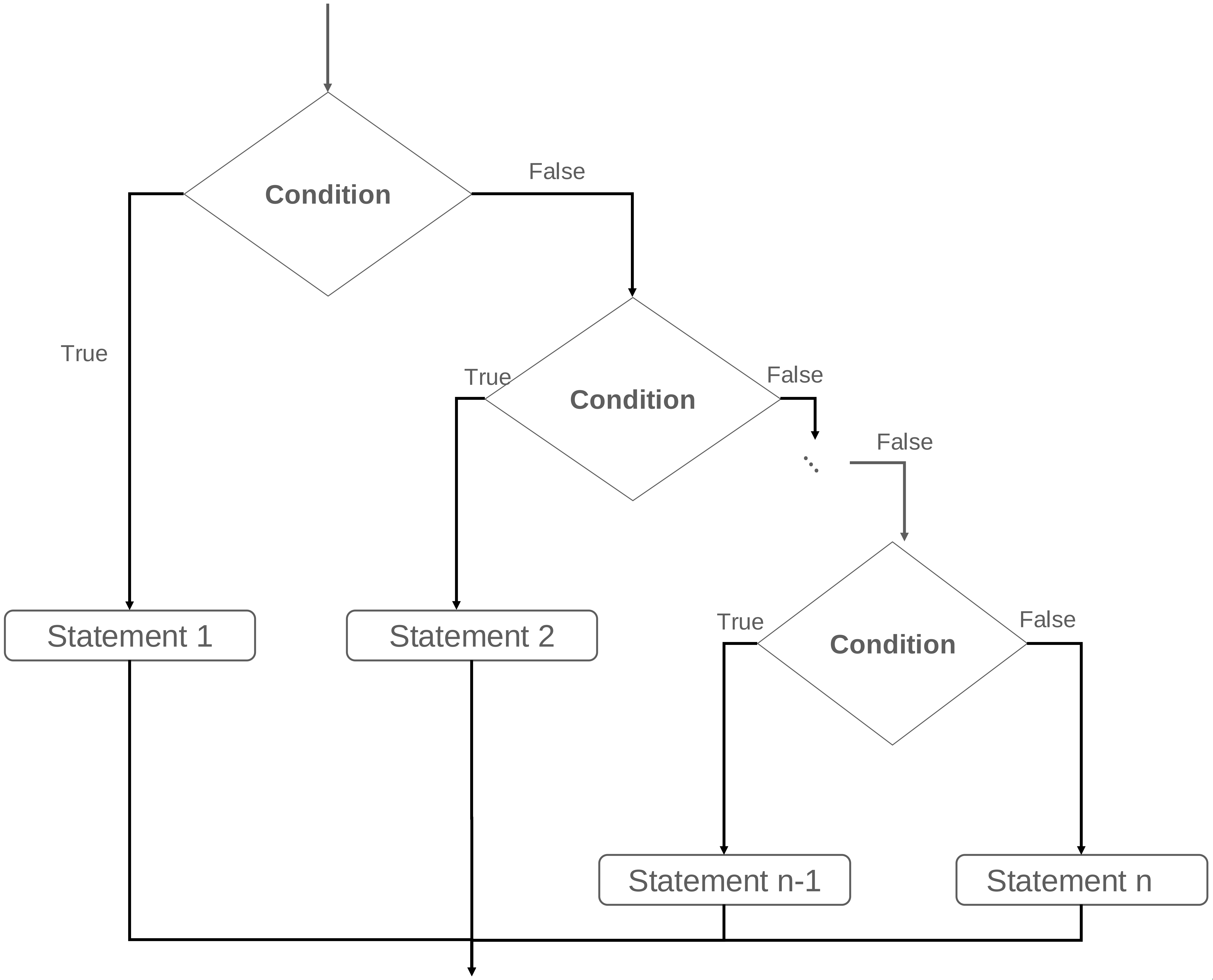
4. **Debugging**

Control flow in Python

1. Decision-making (conditional statements)
   - If
   - match
2. Iteration (Loop statements)
   - while
   - for

If Statements

```
If    condition:
          Statement 1
elif  another condition:
          Statement 2
              ⋮
elif  another condition:
          Statement n-1
else:
          Statement n
```



If statements (example)

$$f(x) = \begin{cases} -1 & x < -1 \\ x & -1 \leq x \leq +1 \\ +1 & x > +1 \end{cases}$$

For other examples go to VS code

Odd-Even numbers

- A number is even if its residual, when devided by 2, is zero!
- Write a program to get an integer number from users and determine if the number is even or odd.

Ternary operator

- Compact coding
- Closer to human language

```
value_if_true if condition else value_if_false
```

Quiz

```
high_income = False

student= True

high_credit = True


if high_income or high_credit or not student:

    print("You are eligible to receive a credit card!")

print("Done!")
```

A - You are eligible to receive a credit card!

B - You are eligible to receive a credit card!
Done!

C – Done!

D – Error!

Match-case statements

```
Match  statement

    Case  1
      Do_something
    Case  2
      Do_something
    Case  3
      Do_something
    Case  4
      Do_something
```

Iteration (loop statements)

- To repeat a task or a piece of code statements
  - Counting
  - Sorting
  - Assessing students’ assignments

Loop types

- while
- for

Special statements

- Break
- Continue

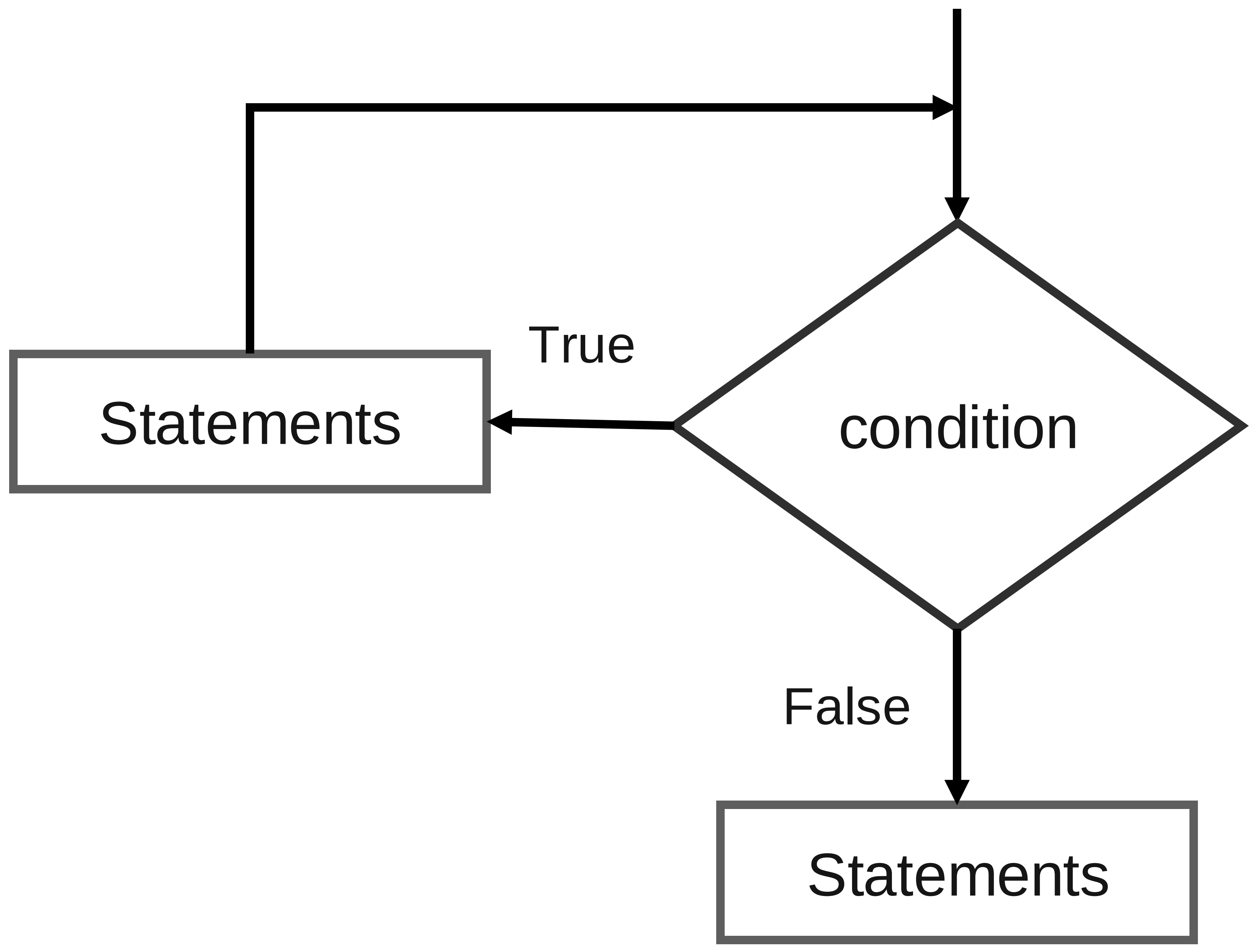
Iterators

zip, enumerate

while loop

Use while to repeat doing something a long as a condition holds

```
while    condition:
    Statements
```



For loop

- Apply repeated actions over a list of items or iterable objects.
- The number of iteration is determined.

```
for   values    in   iterable_objects:
    Statements
```

Iterable objects in python

- List
- Tuple
- String
- range
- Dictionary

Advance looping

1- Nested for

2- Important alert!

3- Indexing: Modifying the iterator during the loop

Nested For Loop

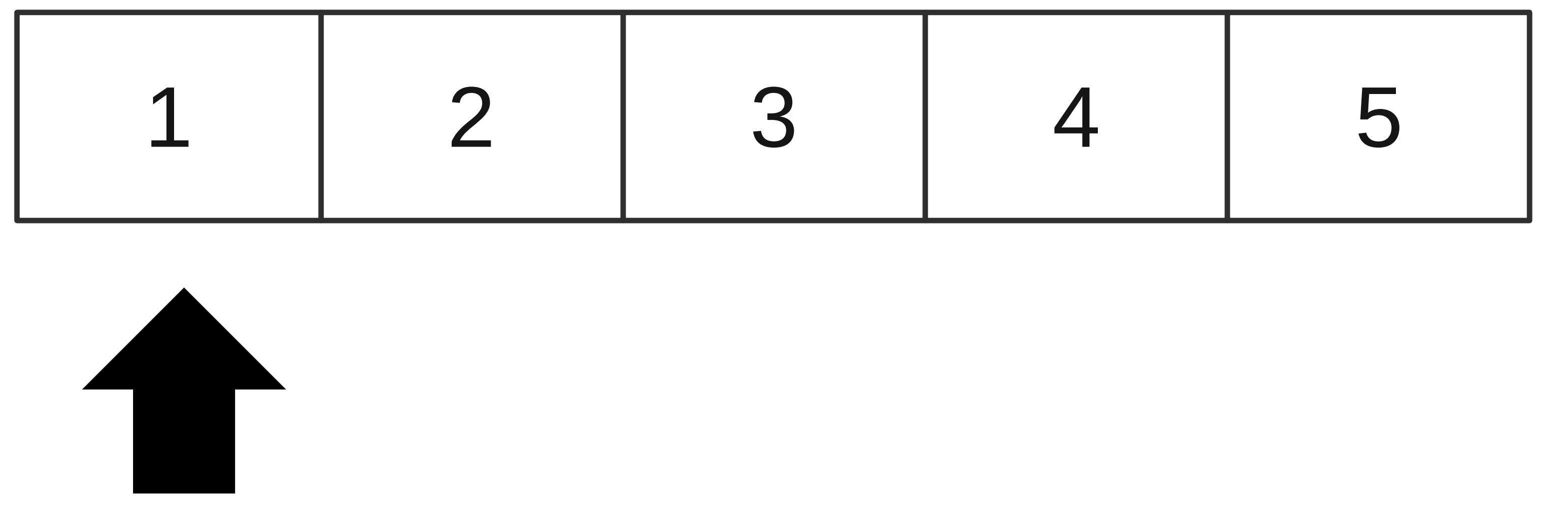
Python allows us to loop inside another loop!

```
for   val1    in   object1:
    for   val2    in   object2:
           Statements
```

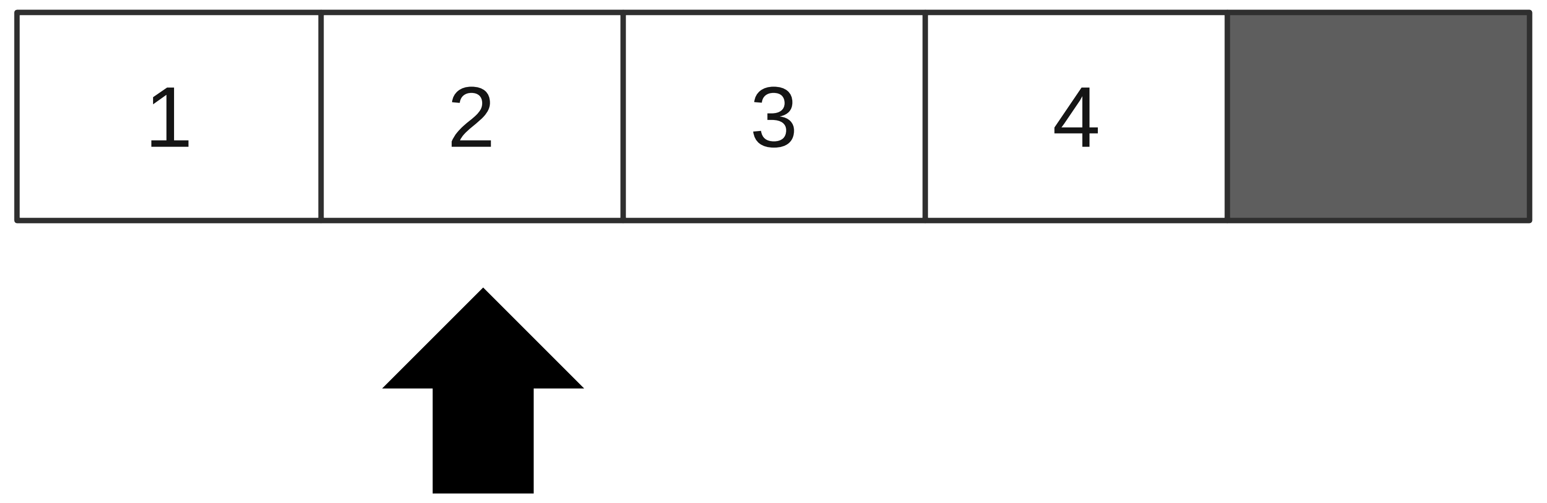
Important alert!

What if the size of the iterable object changes during the for loop?

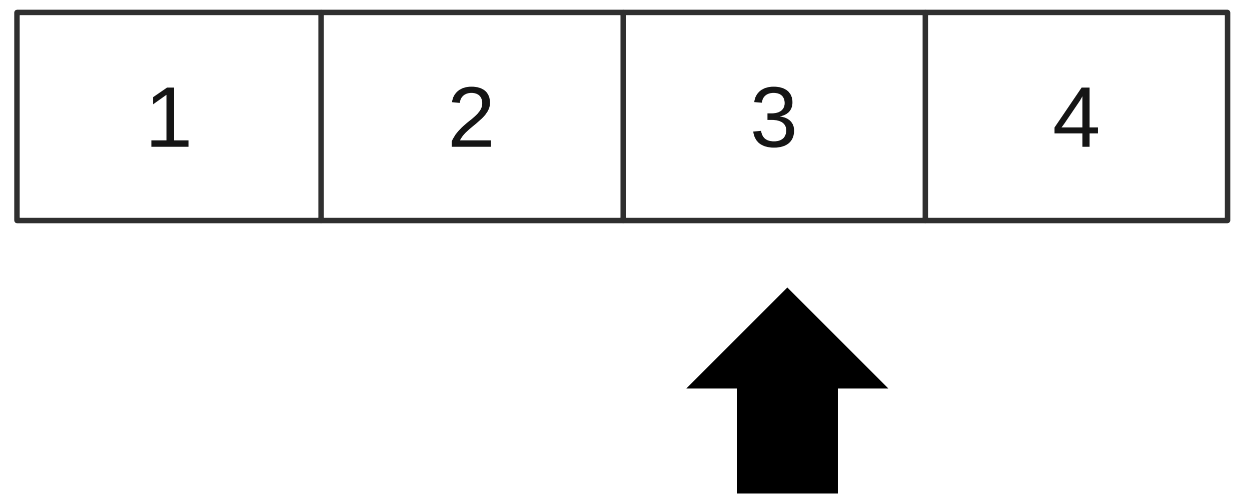
Assume a list of 5 elements



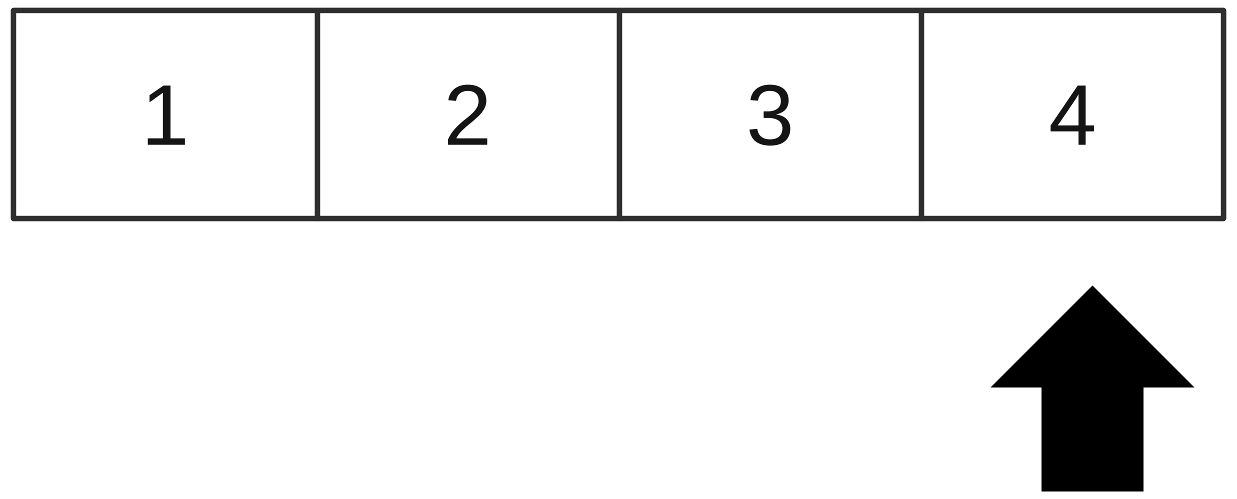
The list's size is reduced by 1 element



The loop keeps processing the next element



The for-loop ends after the 4th iteration



Indexing

- VSCode …

15 Minutes

Break

Enumerate

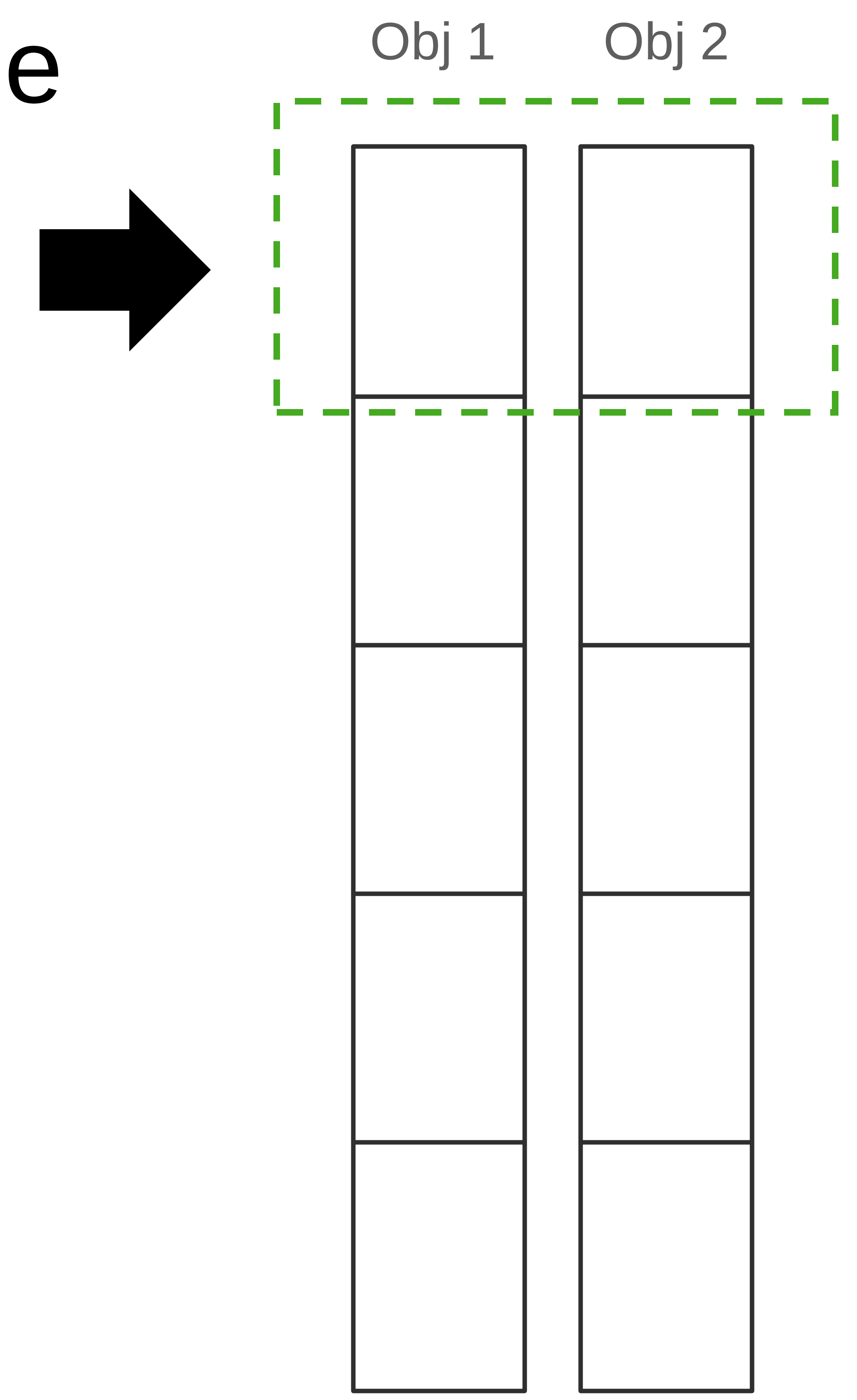
- Associate the sequential order with each elemet of iterable objects

```
for   ind,   val   In  enumerate (object) :

        statements
```

Zip

- Concurrently loop through two lists
- Zip statement combines two iterable objects value-by-value
- Go to vscode



Special statements

Continue & break

- Continue statement ignore the rest of the loop statements in one loop
  - Skip to the next iteartion of the loop
- Break statement ends the entire loop immediately
  - The rest of the loop won’t be executed

List comprehension

- Short syntax for creating a list from another list

Functions in Python

Functions

Types

- Regular definition of functions
- Anonymous functions
- Generators
- Recursive

Function

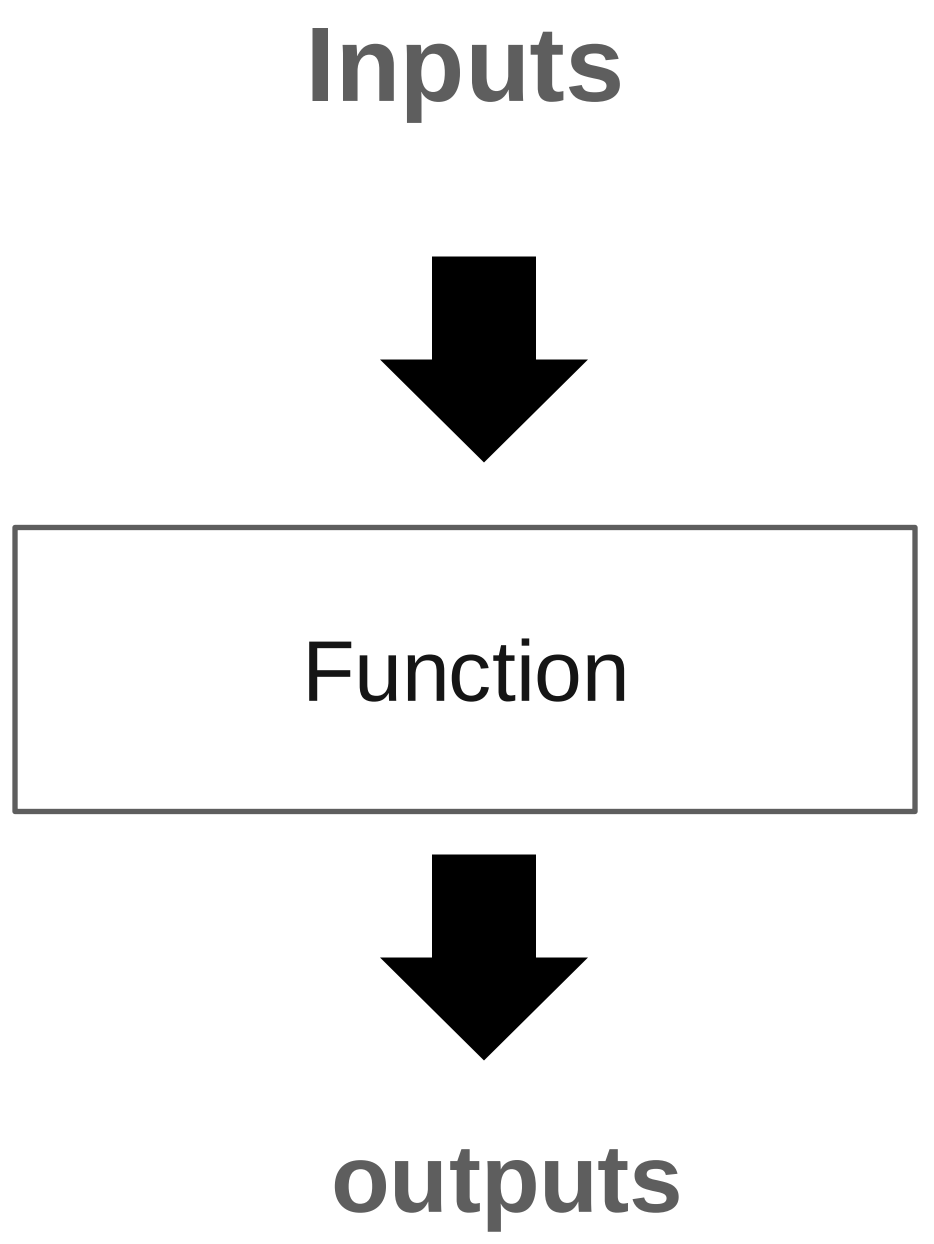
- A block of code that is associated with a name for future references
- Python only execute a function when is being called

```
def    func_name (arguments):

        statements

        return
```

- Arguments : inputs/ variables needed
- Return: outputs values



Anonymous Function

- A function without a name
- A function handler
- Also known as lambda expression or function

```
f = lambda arguments: statements
```

Generator functions

- Allow you to declare a function that behaves like an iterator
- A convenient shortcut to building iterators

```
def    func_name (arguments):

        statements

        yield output
```

Recursive functions

- A function that returns itself!
- Perfect solution for coding dynamic system that their current state depends on their previous states

```
def    func_name (arguments):

    statements

    return func_name(arguments):
```

The try-except statement

- Handle exceptions
- https://docs.python.org/3/library/exceptions.html#concrete-exceptions

```
try:
    statements
except:
    handel exceptions
```

Debaugging (optional)

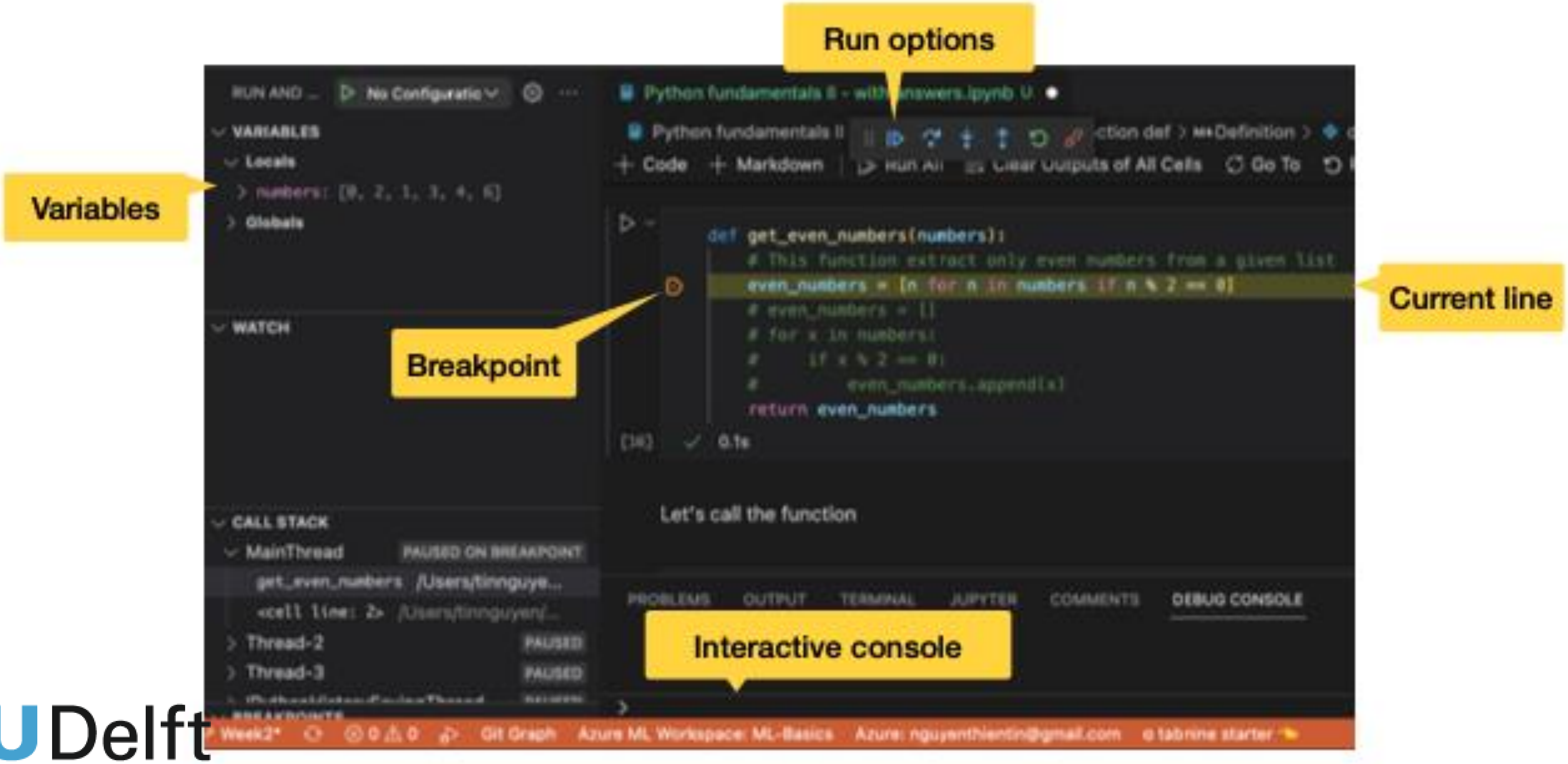
- Use vscode debugger
  - Run code line by line
  - Inspect varibles’s value

Self Study:

https://code.visualstudio.com/docs/python/debugging

Debugging

VSCode



```
numbers: [0, 2, 1, 3, 4, 6]
```

```
MainThread                  PAUSED ON BREAKPOINT
  get_even_numbers  /Users/tinnguye...
  <cell line: 2>  /Users/tinnguyen/...
Thread-2                    PAUSED
Thread-3                    PAUSED
```

```python
def get_even_numbers(numbers):
    # This function extract only even numbers from a given list
    even_numbers = [n for n in numbers if n % 2 == 0]
    # even_numbers = []
    # for x in numbers:
    #     if x % 2 == 0:
    #         even_numbers.append(x)
    return even_numbers
```

```
[16]  ✓  0.1s
```

Let's call the function

Lab session

Lab session

- Open Jupyter Notebook file ‘Fundamental_II_lab_session.ipynb’.
- Update the code in the ‘Exercise’ code blocks.
- Print as pdf
- Hand in on Brightspace
- Deadline: Thursday 12 September