

Information, Computation, Communication

Learning Python

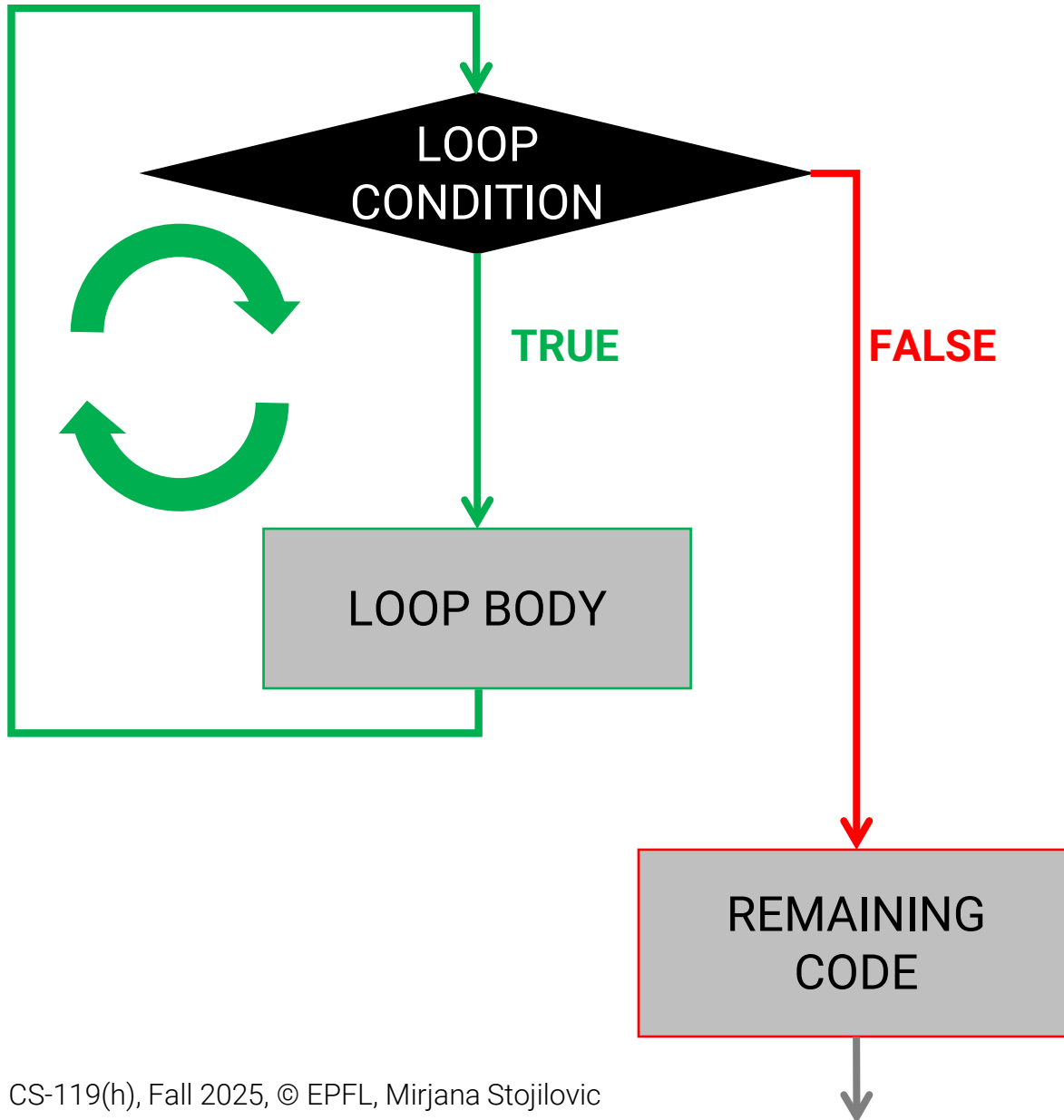
Loops – Part I

Agenda

- [While](#)
 - [Examples](#)
 - [Infinite loops](#)
- [For](#)
 - [Examples](#)
- [range\(\)](#)
- Examples
 - [Factorial](#)
 - [Count digits](#)

while Loops

while Loops



- Python while loop statement **repeatedly** executes a block of code as long as the loop condition evaluates to **true**
- Syntax of the while loop:

```
while condition_is_true:  
    # loop body  
    block_of_code
```

while Loops

Examples

Q: Write a code that, for a given positive number **n**, prints out the sequence of numbers from **0 to n-1**, separated by commas

while Loops

Examples

Q: Write a code that, for a given positive number **n**, prints out the sequence of numbers from **0 to n-1**, separated by commas

```
# Initialize the helper variable i
i = 0
# Loop and print numbers from i to n-1
while i < n:
    # Below, end = is used to specify the characters to print last
    print(i, end=", ")
    # Helper variable must be updated to avoid an infinite loop
    i += 1
```

Infinite Loops



© Guy / Adobe Stock

- Loop condition must be made false at some point during code execution
 - If that never happens, loop repeats infinitely many times
- **Infinite loops are bad**
- In case of unresponsive code or code that does not stop running, use **CTRL-C** combination of keyboard keys to terminate the program execution

while Loops

Example: Infinite Loop

- Try the below lines in the Python interpreter

```
>>> while True:  
...     print('Type Ctrl-C to stop me...')
```

- And type ENTER twice
 - Once to exit the loop
 - Second time to run the code (the loop)

while Loops

Example: Infinite Loop

```
>>> while True:
...     print('Type Ctrl-C to stop me...')
Type Ctrl-C to stop me...
Type Ctrl-C to stop me...
Type Ctrl-C to stop me...
Type Ctrl-C to stop me...
Type Ctrl-C to stop me...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
KeyboardInterrupt
>>>
```

Hit
ENTER
twice here

Hit
Ctrl-C
here



for Loops

for Loops

- In Python, **for** loops step through items in an ordered sequence or an iterable object (strings, lists, tuples, ...)

```
for item in iterable_object:    # assign object item to target
    # loop body
    block_of_code
```

for Loop

Example

Try this example in the Python interpreter

```
>>> for i in 'tictoc':    # assign object item to variable i
...     print(2 * i)      # repeated loop body: use object item
```

for Loop

Example

Try this example in the Python interpreter

```
>>> for i in 'tictoc':    # assign object item to variable i
...     print(2 * i)      # repeated loop body: use object item
...
tt    # The first item is the letter t, repeated twice
ii    # Next item is the letter i, repeated twice
cc    # Next item is the letter c, repeated twice
tt    # Next item is the letter t, repeated twice
oo    # Next item is the letter o, repeated twice
CC    # The last item is the letter c, repeated twice
>>>
```

range()

<https://docs.python.org/3/library/stdtypes.html#range>

Python Function `range()`

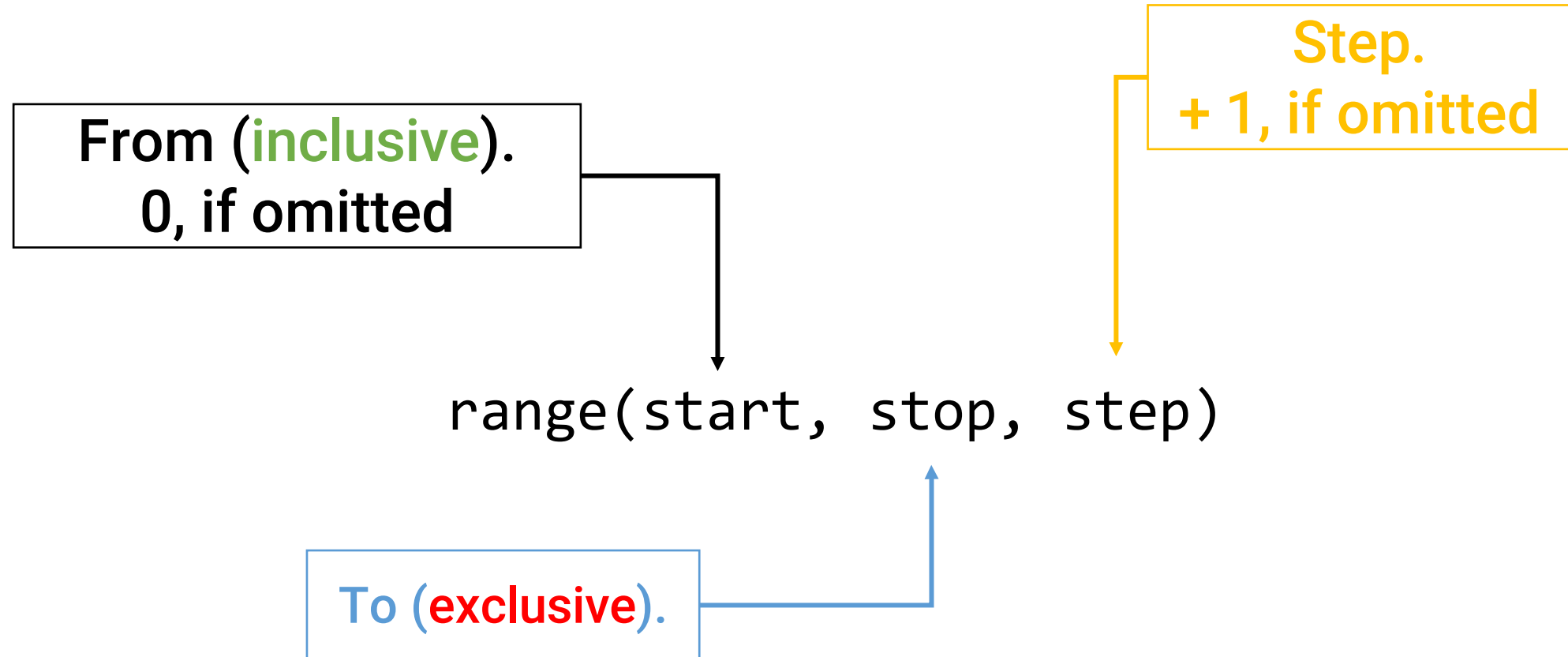
- The Python function `range()` creates a sequence of numbers on the fly
- It can be called in a few different ways:
 - Single parameter form: `range(stop)`
 - Two parameter form: `range(start, stop)`
 - Three parameter form: `range(start, stop, step)`
- To reverse the sequence, use the Python function `reversed()`



<https://docs.python.org/3/library/stdtypes.html#range>

Python Function `range()`

Summary



range(stop)

- **range(stop)** creates a sequence of numbers from **0** to **stop-1**
- We say that **stop** is **exclusive** (not included in the sequence)
- **stop** must be **positive**, or an empty sequence will be returned
- Examples:
 - **range(1)**
 - creates a sequence of a single element: 0
 - **range(5)**
 - creates a sequence of five elements: 0, 1, 2, 3, 4
 - **range(-5)**
 - creates an **empty** sequence (no elements)

range(**start**, stop)

- **range(start, stop)** creates an increasing sequence of numbers from **start** to **stop-1**
- **stop** must be greater than the **start**, or an empty sequence will be returned
- The i^{th} element in the sequence becomes
 - **element_i = start + i**, where $i \geq 0$ and **element_i < stop**
because stop is exclusive

range(start, stop)

- Examples:
 - `range(2, 5)`
 - creates a sequence of three elements: 2, 3, 4
 - `range(-2, 5)`
 - creates a sequence of seven elements: -2, -1, 0, 1, 2, 3, 4
 - `range(-2, -3)`
 - creates an empty sequence (no elements)
 - `range(-2, -1)`
 - creates a sequence of a single element: -2
 - `reversed(range(-1, 1))`
 - returns a sequence 0, -1

range(start, stop, step)

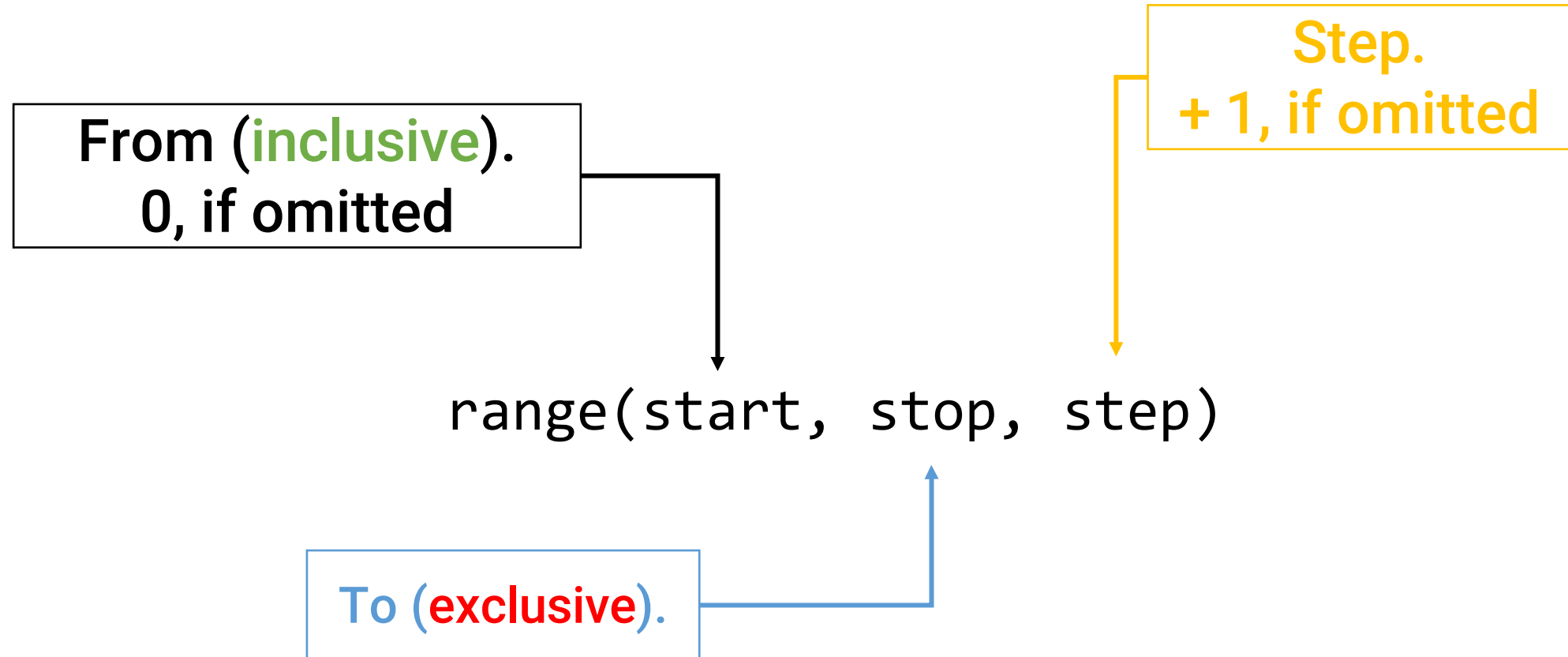
- **range(start, stop, step)** creates a sequence of numbers from **start** to **stop-1**, in steps of **step**
- For a **positive step**, the i^{th} element in the sequence becomes
 - $\text{element}_i = \text{start} + \text{step} * i$, where $i \geq 0$ and $\text{element}_i < \text{stop}$
- For a **negative step**, the i^{th} element in the sequence becomes
 - $\text{element}_i = \text{start} + \text{step} * i$, where $i \geq 0$ and $\text{element}_i > \text{stop}$

range(start, stop, step)

- Examples:
 - `range(2, 5, 2)`
 - creates a sequence of two elements: 2, 4
 - `range(-2, 5, 3)`
 - creates a sequence of three elements: -2, 1, 4
 - `range(-2, -8, 2)`
 - creates an empty sequence (no elements)
 - `range(-2, -8, -2)`
 - creates a sequence of three elements: -2, -4, -6
 - `reversed(range(1, 10, 2))`
 - returns a sequence 9, 7, 5, 3, 1

Python Function `range()`

Summary, Again



Examples

Factorial

Write a piece of code that repeatedly prompts a user to enter an integer number at the input. As long as the received number **n** is negative, the program prompts the user to enter a new number. Otherwise, it computes **n!** and prints the result.

Factorial

Write a piece of code that repeatedly prompts a user to enter an integer number at the input. As long as the received number **n** is negative, the program prompts the user to enter a new number. Otherwise, it computes **n!** and prints the result.

```
ask_for_input = True
while ask_for_input:
    n = int(input("Enter a number..."))
    if n >= 0:
        n_factorial = 1
        for i in range(2, n + 1):
            n_factorial *= i
        print(f"{n}! = {n_factorial}")
        ask_for_input = False
```

Count Digits

Write a piece of code that takes an integer number **v** and computes the number of its digits.

```
# Algorithm to count digits of a positive integer v
digits = 0                                # initialize variable to count the digits
while v:                                  # loop until all digits are processed
    digits += 1                            # increment the digit count for every iteration
    v //= 10                              # remove the last digit
```

Example: before the loop $v = 825$, $digits = 0$

- *after 1st loop iteration: $digits = 1$, $v = 82$*
- *after 2nd loop iteration: $digits = 2$, $v = 8$*
- *after 3rd loop iteration: $digits = 3$, $v = 0$*
- *loop does not restart because $v = 0$*

Count Digits

Extend the program to prompt the user to provide the input number. If the input is zero, the program outputs "Error: Entered number is zero. Exiting..."

```
n = int(input("Enter a number...")) # accept an integer input from the user
if n == 0:
    print("Error: Entered number is zero. Exiting...")
else:
    v = -n if n < 0 else n # make a negative number positive by negating it
    digits = 0             # initialize variable to count the digits
    while v:               # loop until all digits are processed
        digits += 1        # increment the digit count for every iteration
        v //= 10           # remove the last digit
    print(f"There are {digits} digits in {n}.")
```

Count Digits

Without loops; **Python function len()**

Solve the same task, this time without any loop.

*Hint: Python has a built-in function **len()** which returns the number of items in an object (e.g., characters in a string, elements in a list, etc.)*

Count Digits

Without loops

Solve the same task, this time without any loop.

```
n_str = input("Enter a number...") # take the input string (text)
n = int(n_str) # convert it to an integer number

if n: # If n is different than zero
    # If n is positive, the length of the string gives the digit count
    # If n is negative, subtract 1 from the length to account for the negative sign
    digits = len(n_str) if n > 0 else len(n_str)-1
    print(f"There are {digits} digits in {n}.")
else:
    print("Error: Entered number is zero. Exiting...")
```



Next topic: Lists