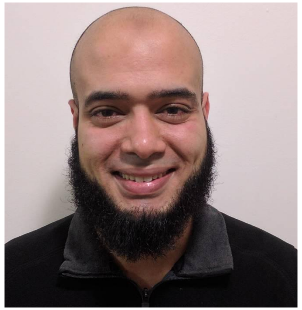
Python Programming
Exceptions

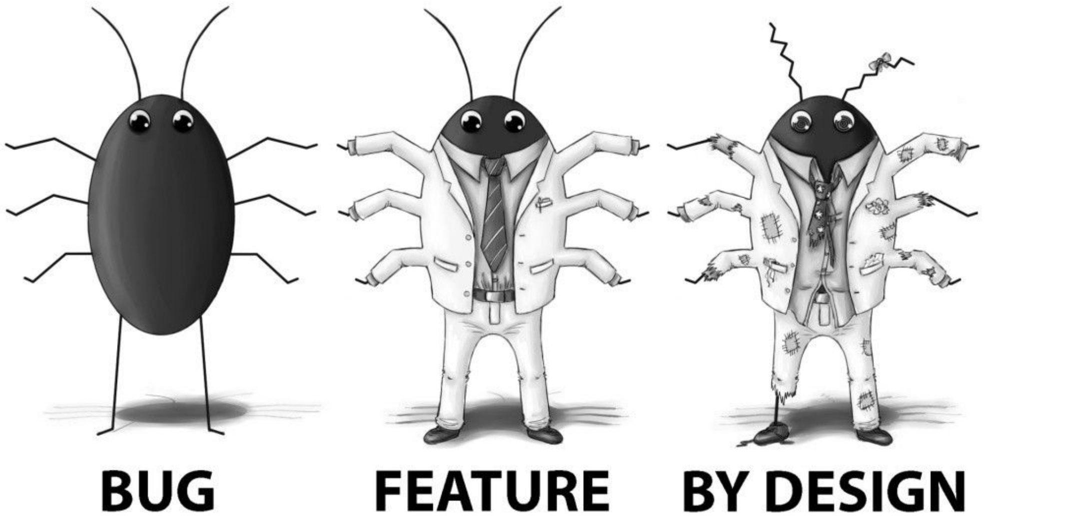
Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Logical errors and Bugs

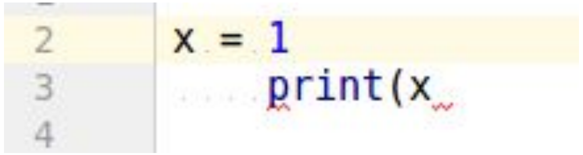
- We create several bugs
- Also users misuse apps
- Or just things go wrong unintentionally



Img src

Syntax Error vs Logical errors

- **Syntax Error**: You did not write the statements in the expected format
  - Parser is complaining. It occurs **before running** the program
  - E.g. Missing parentheses or indentation problem
    - Both of them in line 3 below

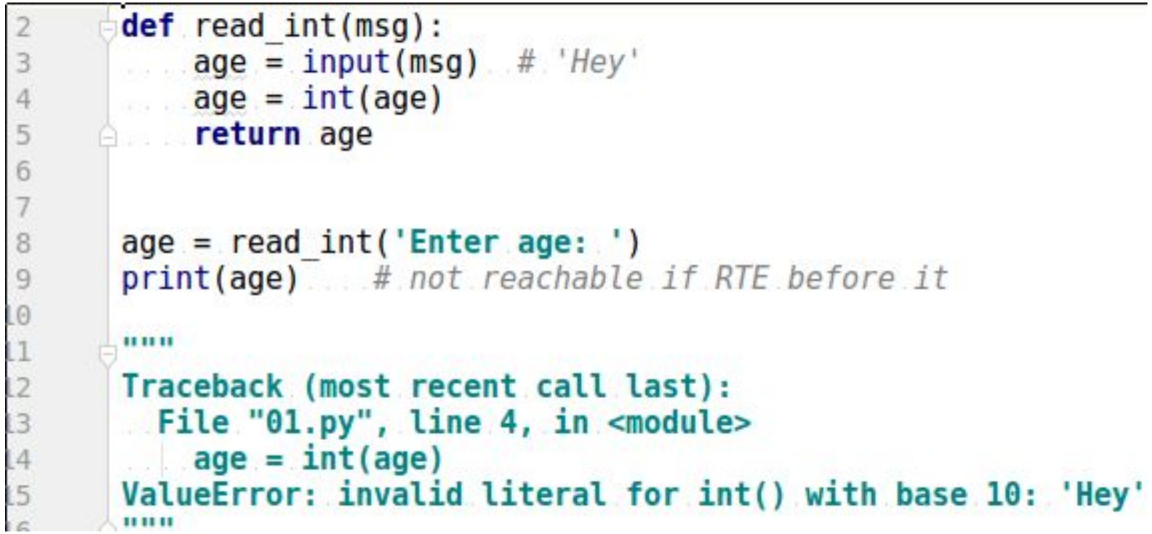


```
2  x = 1
3      print(x
4
```

- **Logical Error**: It occur at runtime (e.g. divide by zero, access invalid index)
  - We call them exceptions!
  - We have to properly handle them!

Logical Error

- We can’t build **production code** this way
- Users will make errors
- Or hackers wanna get service down



```
def read_int(msg):
    age = input(msg)  # 'Hey'
    age = int(age)
    return age


age = read_int('Enter age: ')
print(age)    # not reachable if RTE before it

"""
Traceback (most recent call last):
  File "01.py", line 4, in <module>
    age = int(age)
ValueError: invalid literal for int() with base 10: 'Hey'
"""
```

Blocking Errors

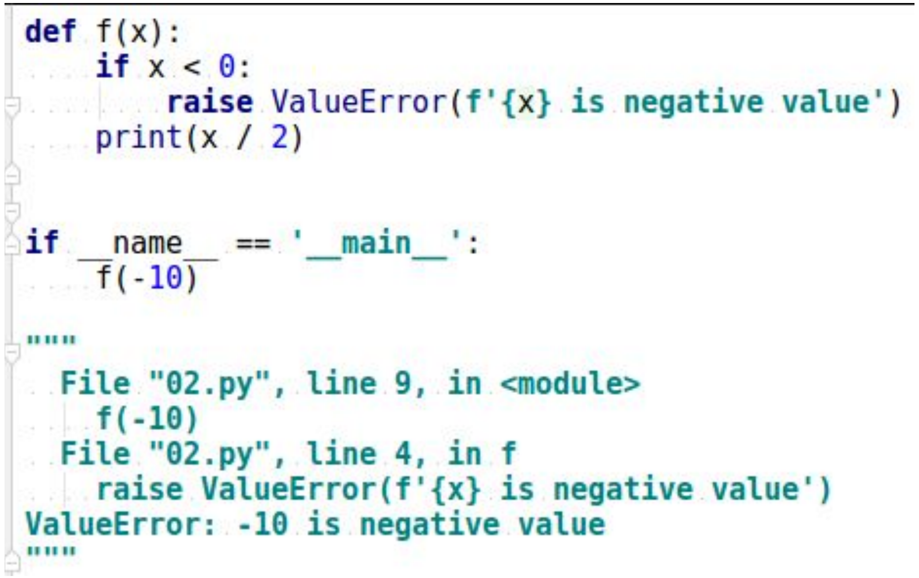
- When we develop applications, we may face conditions where we can’t complete the function
  - Creating an array, but system rejects as no enough memory
  - Open a file, but system rejects due to file permissions
  - Network disconnection during a remote call
  - Payment system: pay a bill, but the money is a negative value!
  - Compute sqrt(x), but x is negative!
  - Coding mistakes: access array out of the boundary
- We typically can’t continue processing. We have to stop!
- Sometimes we can detect the problem, sometimes it just happens!
  - **How** can we communicate as possible the problem? Handle the error?

2 Major approaches

- Return error codes
  - Your function return some number to indicate results
    - E.g. zero for success, 1 for InvalidURL
  - This is not popular python approach.
- **Throwing & Handling Exception**
  - This is a programming language mechanism
    - We can stop processing by raising an exception
    - We can catch it and properly handle it
  - More common & safer
- Future reading: Error codes vs exceptions

Raising Exception

- You can raise errors by yourself
- **ValueError** is one of the built-in classes for exceptions
- Using this syntax you can **raise** exception that can **stop** the code
- Next: we learn how to **handle** the exception to NOT stop our app



```
def f(x):
    if x < 0:
        raise ValueError(f'{x} is negative value')
    print(x / 2)


if __name__ == '__main__':
    f(-10)

"""
  File "02.py", line 9, in <module>
    f(-10)
  File "02.py", line 4, in f
    raise ValueError(f'{x} is negative value')
ValueError: -10 is negative value
"""
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."