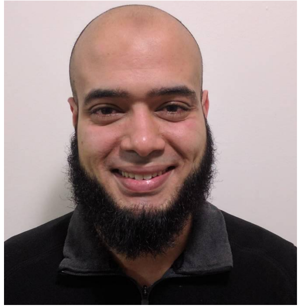
Python Programming

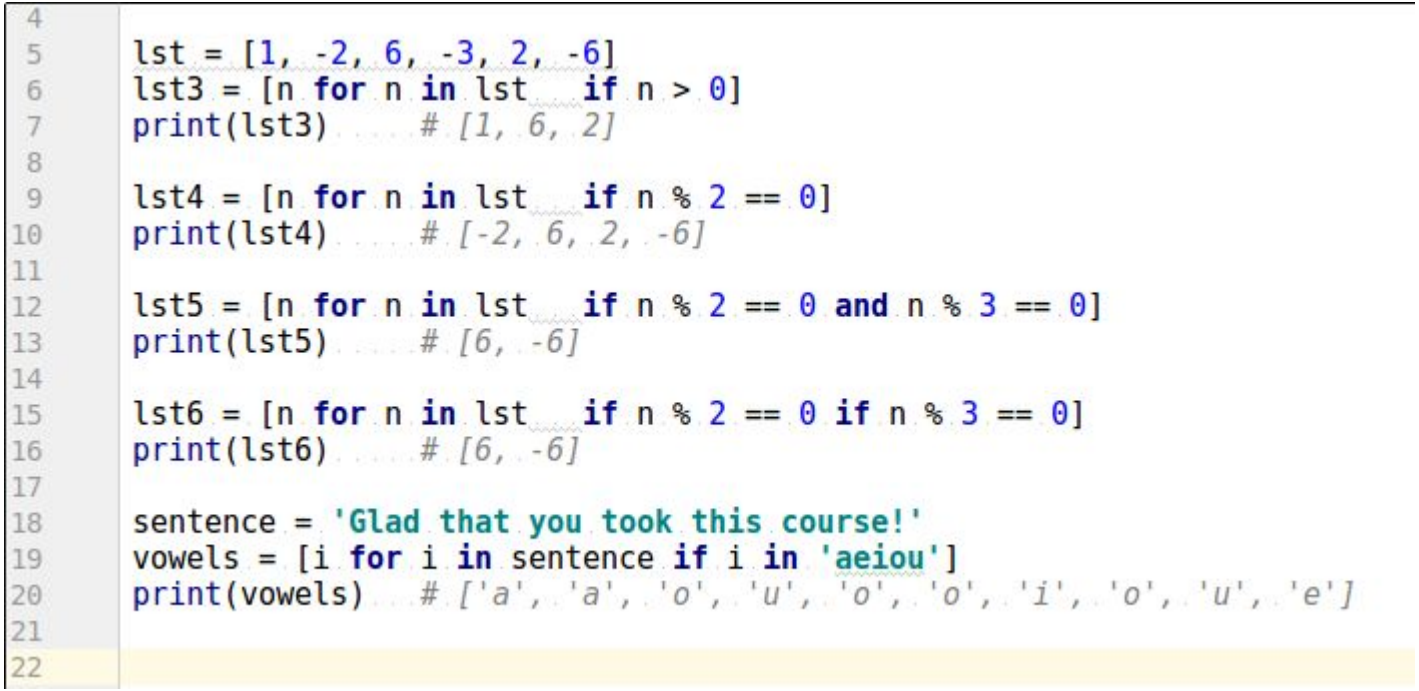
List Comprehension 2

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)

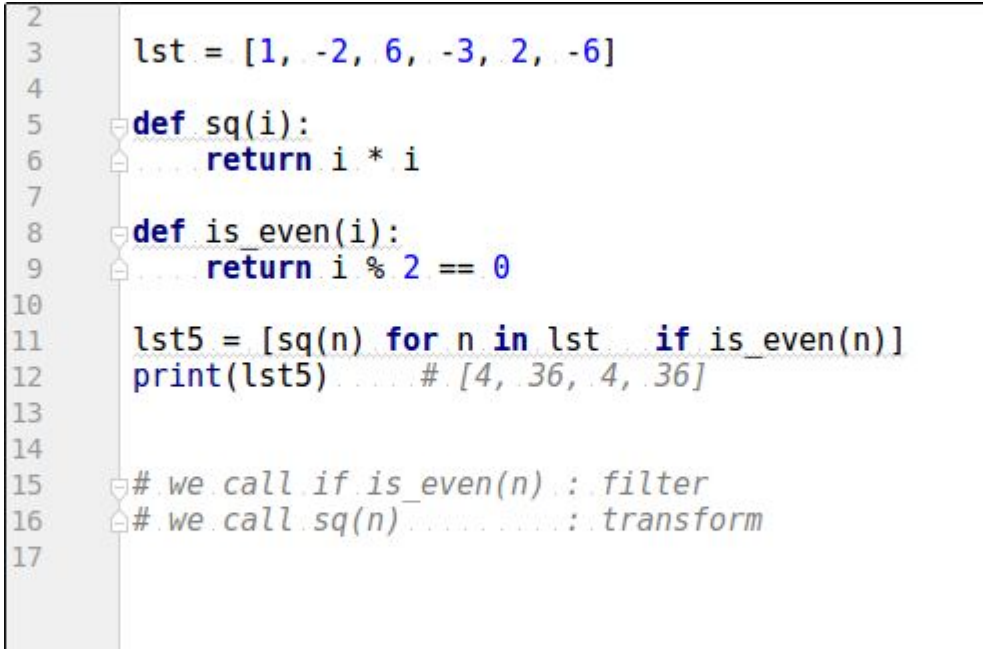


Conditional list comprehension



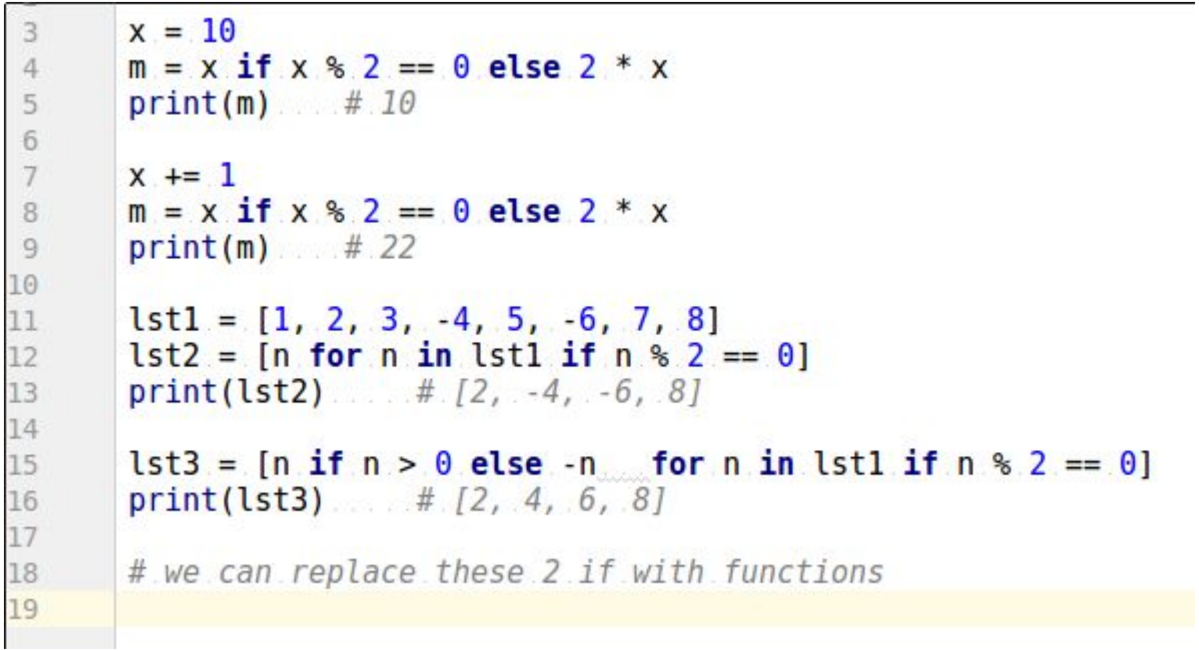
```
lst = [1, -2, 6, -3, 2, -6]
lst3 = [n for n in lst   if n > 0]
print(lst3)     # [1, 6, 2]

lst4 = [n for n in lst   if n % 2 == 0]
print(lst4)     # [-2, 6, 2, -6]

lst5 = [n for n in lst   if n % 2 == 0 and n % 3 == 0]
print(lst5)     # [6, -6]

lst6 = [n for n in lst   if n % 2 == 0 if n % 3 == 0]
print(lst6)     # [6, -6]

sentence = 'Glad that you took this course!'
vowels = [i for i in sentence if i in 'aeiou']
print(vowels)   # ['a', 'a', 'o', 'u', 'o', 'o', 'i', 'o', 'u', 'e']
```

List Comprehension: Filter and Transform



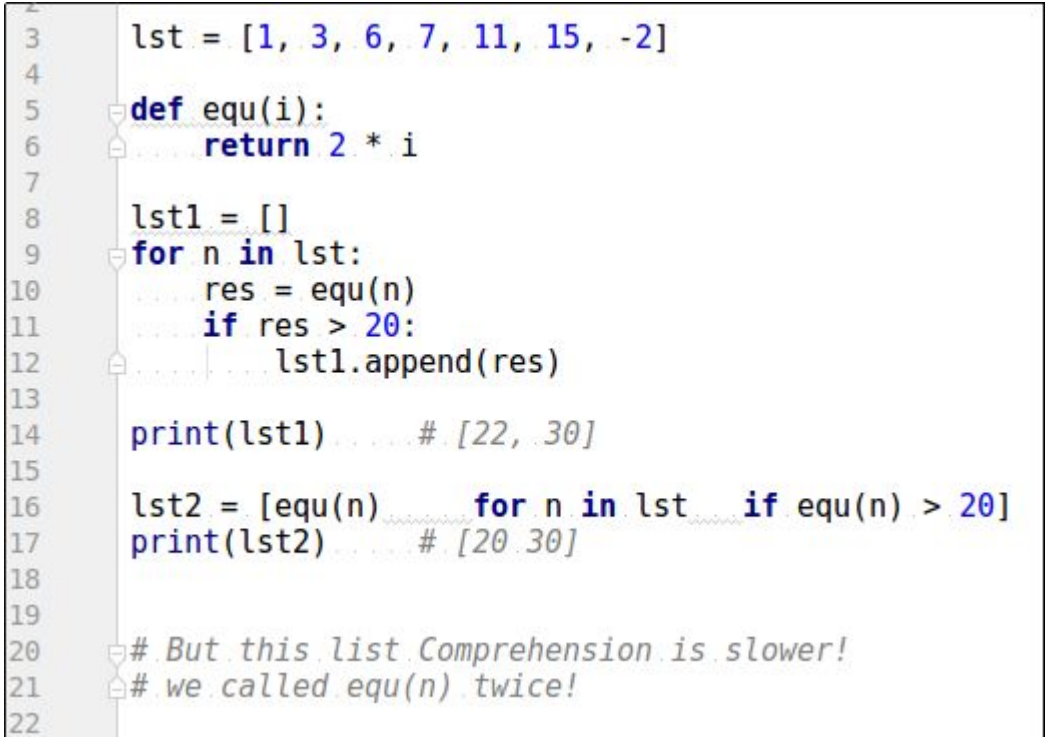
```python

lst = [1, -2, 6, -3, 2, -6]

def sq(i):
    return i * i

def is_even(i):
    return i % 2 == 0

lst5 = [sq(n) for n in lst   if is_even(n)]
print(lst5)     # [4, 36, 4, 36]


# we call if is_even(n) : filter
# we call sq(n)         : transform

```

List Comprehension: if condition: before and after



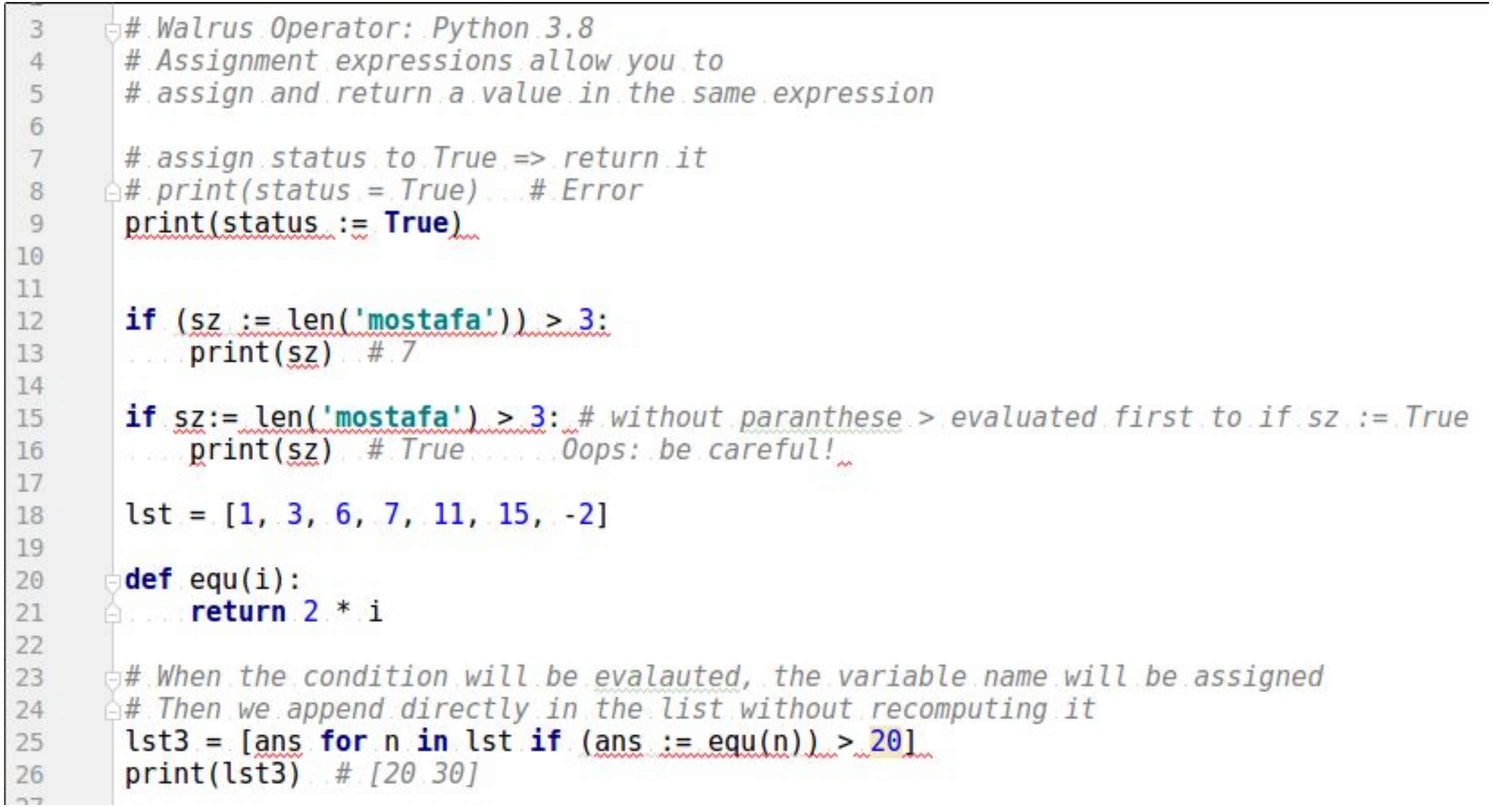
```
x = 10
m = x if x % 2 == 0 else 2 * x
print(m)    # 10

x += 1
m = x if x % 2 == 0 else 2 * x
print(m)    # 22

lst1 = [1, 2, 3,  -4, 5,  -6, 7, 8]
lst2 = [n for n in lst1 if n % 2 == 0]
print(lst2)    # [2,  -4,  -6, 8]

lst3 = [n if n > 0 else -n   for n in lst1 if n % 2 == 0]
print(lst3)    # [2, 4, 6, 8]

# we can replace these 2 if with functions

```

List Comprehension: Slow!



```python
lst = [1, 3, 6, 7, 11, 15, -2]

def equ(i):
    return 2 * i

lst1 = []
for n in lst:
    res = equ(n)
    if res > 20:
        lst1.append(res)

print(lst1)     # [22, 30]

lst2 = [equ(n)     for n in lst   if equ(n) > 20]
print(lst2)     # [20 30]


# But this list Comprehension is slower!
# we called equ(n) twice!
```

Walrus operator



```python
# Walrus Operator: Python 3.8
# Assignment expressions allow you to
# assign and return a value in the same expression

# assign status to True => return it
# print(status = True)    # Error
print(status := True)


if (sz := len('mostafa')) > 3:
    print(sz)  # 7

if sz:= len('mostafa') > 3: # without paranthese > evaluated first to if sz := True
    print(sz)  # True      Oops: be careful!

lst = [1, 3, 6, 7, 11, 15, -2]

def equ(i):
    return 2 * i

# When the condition will be evalauted, the variable name will be assigned
# Then we append directly in the list without recomputing it
lst3 = [ans for n in lst if (ans := equ(n)) > 20]
print(lst3)  # [20 30]
```

“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”