Lecture 12 – EXAMPLES: Prime Numbers

https://cw.fel.cvut.cz/wiki/courses/be5b33prg/start

Michal Reinštein

Czech Technical University in Prague,
Faculty of Electrical Engineering, Dept. of Cybernetics,
Center for Machine Perception
http://cmp.felk.cvut.cz/~reinsmic/
reinstein.michal@fel.cvut.cz

EXAMPLE – PRIME NUMBERS

TASK: *Write a program to generate a list of all prime numbers less than 20*

- Before starting it is important to note what a prime number is:
  - *A prime number has to be a positive integer*
  - *Divisible by exactly 2 integers (1 and itself)*
  - *1 is not a prime number*
- While there are many different ways to solve this problem, here are a few different approaches

SOURCE: https://hackernoon.com/prime-numbers-using-python-824ff4b3ea19

source https://hackernoon.com/prime-numbers-using-python-824ff4b3ea19

EXAMPLE – PRIME NUMBERS

```python
# Approach 1: ForLoops
# Initialize a list
primes = []
for possiblePrime in range(2, 21):

    # Assume number is prime until shown it is not.
    isPrime = True
    for num in range(2, possiblePrime):
        if possiblePrime % num == 0:
            isPrime = False

    if isPrime:
        primes.append(possiblePrime)

```

- Example of a solution

source https://hackernoon.com/prime-numbers-using-python-824ff4b3ea19

EXAMPLE – PRIME NUMBERS

```python
# Approach 1: ForLoops
# Initialize a list
primes = []
for possiblePrime in range(2, 21):

    # Assume number is prime until shown it is not.
    isPrime = True
    for num in range(2, possiblePrime):
        if possiblePrime % num == 0:
            isPrime = False

    if isPrime:
        primes.append(possiblePrime)

```

- **Approach 1:** notice that as soon **isPrime** is False, it is inefficient to keep on iterating. It would be more efficient to **exit out of the loop**

source https://hackernoon.com/prime-numbers-using-python-824ff4b3ea19

EXAMPLE – PRIME NUMBERS

```
15
16  # Approach2: For Loops with Break
17  # Initialize a list
18  primes = []
19  for possiblePrime in range(2, 21):
20
21      # Assume number is prime until shown it is not.
22      isPrime = True
23      for num in range(2, possiblePrime):
24          if possiblePrime % num == 0:
25              isPrime = False
26              break
27
28      if isPrime:
29          primes.append(possiblePrime)
30
```

- **Approach 2:** is more efficient than approach 1 because as soon as you find a given number isn't a prime number you can exit out of loop using break.

source https://hackernoon.com/prime-numbers-using-python-824ff4b3ea19

EXAMPLE – PRIME NUMBERS

```python

# Approach 3: For Loop, Break, and Square Root
# Initialize a list
primes = []
for possiblePrime in range(2, 21):

    # Assume number is prime until shown it is not.
    isPrime = True
    for num in range(2, int(possiblePrime ** 0.5) + 1):
        if possiblePrime % num == 0:
            isPrime = False
            break

    if isPrime:
        primes.append(possiblePrime)

```

- **Approach 3:** is similar to the approach 2 except the inner range function. Notice that the inner range function is now:

  range(2, int(possiblePrime ** 0.5) + 1)

source https://hackernoon.com/prime-numbers-using-python-824ff4b3ea19

EXAMPLE – PRIME NUMBERS

- We use the properties of **composite numbers**
- Composite number is a **positive** number **greater than 1** that is **not prime** (which has factors other than 1 and itself)
- Every composite number has a factor **less than or equal to its square root** (proof here).

- **EXAMPLE**: *Factors of 15 below; the factors in red are just the reverses of the green factors so by the commutative property of multiplication 3 x 5 = 5 x 3 we just need to include the "green" pairs to be sure that we have all the factors.*

Factors of 15				
Factor 1	1	3	5	15
Factor 2	15	5	3	1

source https://hackernoon.com/prime-numbers-using-python-824ff4b3ea19

EXAMPLE – PRIME NUMBERS

```
55  import timeit
56
57  # Approach 1: Execution time
58  print(timeit.timeit('approach1(500)', globals=globals(), number=100000))
59  # Approach 2: Execution time
60  print(timeit.timeit('approach2(500)', globals=globals(), number=100000))
61  # Approach 3: Execution time
62  print(timeit.timeit('approach3(500)', globals=globals(), number=100000))
63
```

- Evaluating performance

- **REFERENCE:** https://hackernoon.com/prime-numbers-using-python-824ff4b3ea19

source https://hackernoon.com/prime-numbers-using-python-824ff4b3ea19