|  |  |  |
| --- | --- | --- |
| **College of Engineering**  Computer Science & Eng. Dept.  **Course:** CMP 321L Programminglanguages Lab | A picture containing logo  Description automatically generated | **Course Professor:** Dr. Michel Pasquier  **Lab Instructor:** Praveena Kolli  **Office:** EB2-126  **Phone**: 971-6-5152352  **e-mail**: pkolli@aus.edu  **Semester**: Summer 2022 |

**Lab 6 – iterators and generators**

**Objectives:**

* Define functions with iterators and generators

**Due date: End of the lab. (**Only one team member needs to submit.)

**Rules:**

(1) Usage: **You should explore and make good use of the Python features you learned in class.** (2) Scope: **You should only use those features that have been explained in detail in class.**

(3) Style: Follow standard Python programming style and conventions.

(4) Logic: Add appropriate comments to your code to explain your solution.

(Code / answers that do not follow the above specifications will be penalized.)

***Warning:* You need to come to the lab properly prepared i.e.**

(1) Make sure you have studied and understood the class material.

(2) Read the lab doc, think about the problems, and prepare questions as needed.

If you do not, completing the lab in 2.45 hours may become too much of a challenge!

**Useful resources:**

<https://docs.python.org/3/library/itertools.html>

[**https://wiki.python.org/moin/Generators**](https://wiki.python.org/moin/Generators)

**Lab By:**

Rohan Mitra 85023

Keshav Ramesh 88595

**Github link for code**: <https://github.com/ro1406/CMP321Lab>

Recall that *n* is a **prime** if it is divisible only by 1 and itself, hence *not* divisible by any integer in the range from 2 to *sqrt(n),* both inclusive.

**Exercise 1: recursive function [1 mark]**

Write a Python *recursive* *function* that checks if positive integer *n* is a prime.

**Code:**

#Q1:

import math

def isPrime(n,div=2):

if n<=2:

return n==2

if div>math.sqrt(n):

return True

if n%div==0:

return False

return isPrime(n,div+1)

if \_\_name\_\_ == '\_\_main\_\_':

for x in range(2,30+1): print(f'{x}: {isPrime(x)}')

**Screenshot:**

**Text

Description automatically generated**

**Exercise 2: iterator class [4 marks]**

1. Create a Primes *iterator* class that will give prime numbers infinitely.
2. Write an initializer that takes a starting number, give a default value of 2.
3. Write the \_\_iter\_\_ method that returns the object itself.
4. Write the \_\_next\_\_ method that produces the next prime number.
5. Test the class developed in ex2 part(a)
6. Write a for loop to print the first 20 primes.
7. Use list comprehension to create a list of the next 20 primes.
8. Use islice() to create a list of the first 20 primes larger than 1 million.

**Code:**

from Q1 import isPrime

class Primes:

def \_\_init\_\_(self,s=2):

self.curr=s

def \_\_iter\_\_(self):

return self

def \_\_next\_\_(self):

n=self.curr

while not isPrime(n):

n+=1

self.curr=n+1

return n

#1:

primes=Primes()

print("First 20 primes with a for loop:")

for x in range(20):

print(next(primes))

print('-'\*95)

#2:

primes=Primes()

twentyPrimes=[next(primes) for x in range(20)]

print("First 20 primes using list comprehension:")

print(twentyPrimes)

print('-'\*95)

#3:

from itertools import islice

primes=Primes(1\_000\_000)

print("First 20 primes over 1 million using islice:")

print(list(islice(primes,0,20)))

**Screenshot:**

Graphical user interface, text, application

Description automatically generated

**Exercise 3: generator function [5 marks]**

1. Write a prime number *generator*, using the prime function developed in ex1. By default it should generate (infinitely) *all* primes. If given a value as argument, it should generate (infinitely) all primes larger than this value.
2. Test the class developed in ex3 part(a)
3. Write a for loop to print the first 20 primes.
4. Use list comprehension to create a list of the next 20 primes.
5. Use islice() to create a list of the first 20 primes larger than 1 million.
6. Use a generator expression to create an iterator that would produce the first 100 primes larger than 1 million, then use a for loop and this iterator to print the values.
7. Use the above iterator to create a list of the first 100 primes larger than 1 million for which the last two digits are identical.

**Code:**

#part (a)

from Q1 import isPrime

def generatePrimes(x=2):

while True:

if isPrime(x):

yield x

x+=1

#part(b)

#1:

print("First 20 primes with a for loop:")

gen=generatePrimes()

for x in range(20):

print(next(gen), end = " ")

print('-'\*95)

#2:

twentyPrimes=[next(gen) for x in range(20)]

print("First 20 primes using list comprehension:")

print(twentyPrimes)

print('-'\*95)

#3:

from itertools import islice

print("First 20 primes over 1 million using islice:")

print(list(islice(generatePrimes(1\_000\_000),0,20)))

print('-'\*95)

#4:

gen=generatePrimes(1\_000\_000)

print("First 100 primes past a million: ")

for x in range(100):

print(next(gen), end = " ")

print('-'\*95)

#5:

gen=generatePrimes(1\_000\_000)

print("The list of primes after 1 million with identical last digits are: ")

print([x for x in islice(generatePrimes(1\_000\_000),0,100) if str(x)[-1]==str(x)[-2]])

**Screenshot:**

Graphical user interface, text, application

Description automatically generated