IIT Madras
BSc Degree

Home Week-2

PDSA - Week 1

For Python

https://pypod.github.io/

Number of operation or instruction when program runs

Count the number of operation or instruction executed when program runs.

```python
def total(a,b):
    s = a + b
    return s
```

```python
n = int(input())
s = 0
for i in range(n):
    s = s + i
print(s)
```

```python
def total(n):
  s = 0
  for i in range(n):
    for j in range(n):
      s = s + 1
  return s
```

Identify the relationship between number of instructions and input size.

Our Goal - Want to reduce these numbers of instructions when program runs

Computing `gcd`

- `gcd(m, n)` **— greatest common divisor**
  - Largest k that divides both m and n
  - `gcd(8, 12)` = 4
  - `gcd(18, 25)` = 1
- **Also** `hcf` **— highest common factor**
  - `gcd(m, n)` always exists
  - 1 divides both m and n
- **Computing** `gcd(m, n)`
  - `gcd(m, n) ≤ min(m, n)`
  - Compute list of common factors from 1 to `min(m, n)`
  - Return the last such common factor

```python
def gcd(m,n):
    cf = [] # List of common factors
    for i in range(1,min(m,n)+1):
        if (m%i) == 0 and (n%i) == 0:
            cf.append(i)
    return(cf[-1])
```

Computing `gcd` - Eliminate the list

```python
def gcd(m,n):
    for i in range(1,min(m,n)+1):
        if (m%i) == 0 and (n%i) == 0:
            mrcf = i
    return(mrcf)
```

Efficiency :- Both versions of `gcd` take time proportional to `min(m, n)`

Computing `gcd` - Better Way

- Suppose d divides m and n
- m = ad, n = bd
- m − n = (a − b)d
- d also divides m − n

```python
def gcd(m,n):
    (a,b) = (max(m,n), min(m,n))
    if a%b == 0:
        return(b)
    else
        return(gcd(b,a-b))
```

Still not efficient, for example `gcd(1,1000)` takes 1000 steps.

Computing `gcd` - Euclid's algorithm

- If n divides m, gcd(m, n) = n
- Otherwise, compute gcd(n, m mod n)

```python
def gcd(m,n):
    (a,b) = (max(m,n), min(m,n))
    if a%b == 0:
        return(b)
    else
        return(gcd(b,a%b))
```

Can show that this takes time proportional to number of digits in `max(m, n)`

Computing `Prime`

- A prime number n has exactly two factors, 1 and n
- Note that 1 is not a prime
- Compute the list of factors of n
- n is a prime if the list of factors is precisely `[1,n]`

```python
def factors(n):
    fl = [] # factor list
    for i in range(1,n+1):
        if (n%i) == 0:
            fl.append(i)
    return(fl)
def prime(n):
    return(factors(n) == [1,n])
```

Counting primes

```python
def primesupto(m):
    pl = [] # prime list
    for i in range(1,m+1):
        if prime(i):
            pl.append(i)
    return(pl)
```

Computing Primes- Other approach

- **Directly check if n has a factor between 2 and n − 1**

```
def prime(n):
    result = True
    for i in range(2,n):
        if (n%i) == 0:
            result = False
    return(result)
```

- **Directly check if n has a factor between 2 and n//2**

```
def prime(n):
    result = True
    for i in range(2,n//2):
        if (n%i) == 0:
            result = False
    return(result)
```

- **Terminate after we find first factor**

```
def prime(n):
    result = True
    for i in range(2,n):
        if (n%i) == 0:
            result = False
            break # Abort loop
return(result)
```

Computing Primes- Sufficient to check factors up to √ n

```
import math
def prime(n):
    (result,i) = (True,2)
    while (result and (i <= math.sqrt(n))):
        if (n%i) == 0:
            result = False
        i = i+1
    return(result)
```

Exception handling

Our code could generate many types of errors

- y = x/z, but z has value 0
- y = int(s), but string s does not represent a valid integer
- y = 5*x, but x does not have a value
- y = l[i], but i is not a valid index for list l
- Try to read from a file, but the file does not exist
- Try to write to a file, but the disk is full

Types of some common errors

- `SyntaxError: invalid syntax`
- Name used before value is defined - `NameError: name ’x’ is not defined`
- Division by zero in arithmetic expression - `ZeroDivisionError: division by zero`
- Invalid list index `IndexError: list assignment index out of range`

Handling exceptions

```
try:
    #... ← Code where error may occur
except (IndexError):
    #... ← Handle IndexError
except (NameError,KeyError):
    #... ← Handle multiple exception types
except:
    #... ← Handle all other exceptions
else:
    #... ← Execute if try runs without errors
```

Classes and objects

Abstract datatype

- Stores some information
- Designated functions to manipulate the information
- For instance, stack: last-in, first-out, push(), pop()
- Separate the (private) implementation from the (public) specification

Class

- Template for a data type
- How data is stored
- How public functions manipulate data

Object

- Concrete instance of template

Example

```
class Point:
  def __init__(self,a=0,b=0):
    self.x = a
    self.y = b

  def translate(self,deltax,deltay):
    self.x += deltax
    self.y += deltay

  def odistance(self):
    import math
    d = math.sqrt(self.x*self.x +
                  self.y*self.y)
    return(d)

```

```python
  def __str__(self):
    return('('+str(self.x)+','
            +str(self.y)+')')

  def __add__(self,p):
    return(Point(self.x + p.x,
                 self.y + p.y))

p = Point(3,4)
q = Point(5,8)
print(p)
print(p+q)
```

Output

```
(3,4)
(8,12)
```

Timer

```python
import time

class TimerError(Exception):
    """A custom exception used to report errors in use of Timer class"""

class Timer:
    def __init__(self):
        self._start_time = None
        self._elapsed_time = None

    def start(self):
        """Start a new timer"""
        if self._start_time is not None:
            raise TimerError("Timer is running. Use .stop()")
        self._start_time = time.perf_counter()

    def stop(self):
        """Save the elapsed time and re-initialize timer"""
        if self._start_time is None:
           raise TimerError("Timer is not running. Use .start()")
        self._elapsed_time = time.perf_counter() - self._start_time
        self._start_time = None

    def elapsed(self):
        """Report elapsed time"""
        if self._elapsed_time is None:
           raise TimerError("Timer has not been run yet. Use .start()")
        return(self_elapsed_time)

    def __str__(self):
        """print() prints elapsed time"""
        return(str(self._elapsed_time))
```

Set recursion limit

```python
import sys
sys.setrecursionlimit(100000)
gcd(2,99999)
```

Calculate time for large value

```python
# 10^16
t.start()
print(678912345678912345,987654321987654321,gcd(678912345678912345,987654321987654321))
t.stop()
print(t)
```

Why Efficiency?

Example-

- Sort all Aadhaar number
- Search data from big database
- Real time Gamming Problem