University College Cork
Computer Science
Department

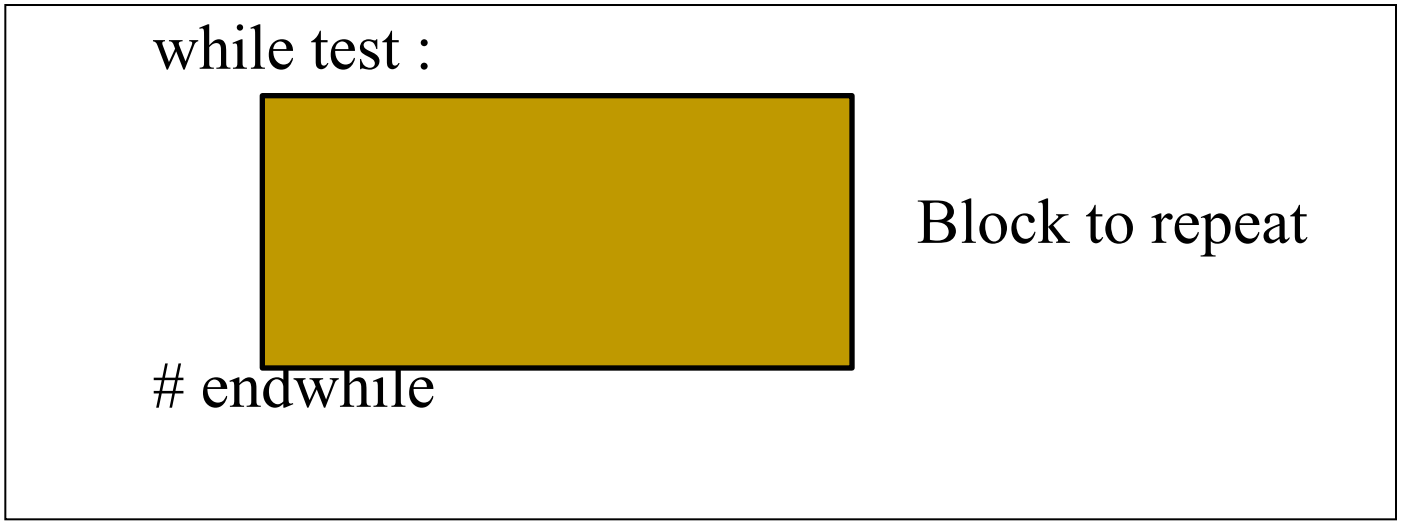
Programming Training

Main Points:

- Working with Functions in Python
- Problems with Numbers.
- Discuss some important algorithms
  - Primality
  - Digits
  - Greatest Common Divisor

Python Repetitions.

while repetitions execute repeatedly a statement while a condition holds or for a number of times.



While loop repeats the block while test holds. If test is initially false the the block is not exeecuted.

Python Repetitions.

for repetitions execute repeatedly a statement while a condition holds or for a number of times.

for elem in iterator :

Block to repeat

endfor

For loop repeats statements for all elems in iterator.

iterator is a set of elements that can be traversed.

Python 3.4.1: decompositionpower.py – /Users/sabin/Desktop/python/decompositionpower.py

```
# import modules
import math

def decomposition():

    # input n d
    n = int(input('n='))

    #traverse the possible divisors

    for d in range(2,n+1) :

        if n % d == 0 :

            # count how many times d can be extracted n
            count = 0

            while n % d == 0 :

                count=count+1
                n = n/d

            #endwhile

            print(d, '**' ,count)

        # endif

    # endfor

#end main
```

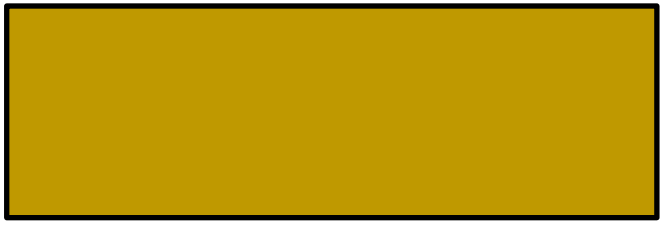
Functions in Python

A Python program is a sequence of a functions which can be executed.

Function = Routine = Procedure = Method = Solution for a sub-problem.

A Python function is written once and used/called as many times as needed.

```
def function_name(arg1, arg2,…):
```



```
# end function
```

1. function_name is the name of the function
2. arg1, arg2,…are the function arguments (usually the inputs)
3. the function body must find the output and return it.

Steps to work with functions

1. Define these functions you want to work with

- Start from the arguments as inputs.
- Give the sequence of statements to construct the output res.
- Return the output res.

2. Call a function at any time you need.

- The call is function_name(a1,a2,…) and returns a value that can be used.
- a1, a2,… are the actual/current arguments

3. Actual arguments are NOT change after the call even if they change in the function

Example 1.

1. Find the minimum and maximum of two integers a and b.

Inputs: a, b – int

Outputs: max, min – int

How to do it? Compare a and b and then decide on min and max

```
def maxMin(a, b):
        # compare a and b
        if a >= b :
            max = a
            min = b
        else :
            max = a
            min = b
        # endif
        return max, min
# end maxMin
```

How to find the max of 4 numbers a, b, c, d

```
def max4Nrs(a, b, c, d):

        max1, min1 = maxMin(a,b)
        max2, min2 = maxMin(c,d)
        max3, min3 = maxMin(max1, max2)

        return max3

# end max4Nrs
```

max4nrs - C:/Users/stabirca/Desktop/python/max4nrs

File Edit Format Run Options Windows Help

```
import math

def toExecute():

    # read inputs
    a, b, c, d = map(int, input('input a b c d: ').split())
    # calculate the overall max
    maximum = max4Nrs(a, b, c, d)
    # print it
    print('maximum = ', maximum)

# end toExecute

def maxMin(a, b):
    # compare a and b
    if a >= b :
        max = a
        min = b
    else :
        max = b
        min = a
    # endif

    return max, min

# end maxMin

def max4Nrs(a, b, c, d):

    max1, min1 = maxMin(a,b)
    max2, min2 = maxMin(c,d)
    max3, min3 = maxMin(max1, max2)

    return max3

# end max4Nrs
```

Ln: 13 Col: 0

Find pow so that d^pow | n.

Any time when a divisor is found we remove it from the number.

Counting problems use a counting counter cont:

- initial value is 0.
- count increases when the event occurs.

while n is divisible by d

- **divide n by d**
- **count that**

Find pow so that d^pow | n.

Inputs: n, d – long ➔ The inputs are the function arguments

Output: pow – long ➔ The output must be calculated and returned.

Remarks:

1) The call power(n,d) does not change n

2) The function can be applied in prime decomposition

- traverse all possible divisors
- find the power of one divisor
- extract it from the number

```
def power(n,d):
    count = 0

    while n % d == 0 :

        # intiger division and count
        n = n // d
        count = count + 1

    # endwhile

    return count
# end power
```

primedecomposition - C:/Users/stabirca/Desktop/python/primedecomposition

File Edit Format Run Options Windows Help

```
import math

def power(n,d):
    count = 0

    while n % d == 0 :

        # intiger division and count
        n = n // d
        count = count + 1

    # endwhile

    return count
# end power

def primeDecompose(n):

    # traverse the divisors
    for d in range(2,n+1) :
        powD = power(n, d)
        if powD > 0 :
            # print d and powD
            print(d, ' ** ', powD)
            # remove the factor from n
            n = n // (d ** powD)
        # endif
    #endfor
# end primeDecompose

def toExecute():
    n = int(input('input n'))

    primeDecompose(n)

# end toExecute
```

Ln: 36 Col: 0

Prime numbers

If a number n is integer then find whether is prime or not.

n is prime when n has only 2 divisors 1 and n.

Example:

- n =2, 3, 5, 7 are all prime
- even numbers are not prime.
- 123456789109876543 21 is prime
- -1234567891010987654321 is prime too

Prime numbers (1)

Some Remarks:

n==2 ➔ n is prime.

Even numbers: n%2==0 ➔ n is not prime

Odd numbers have only odd divisors d=3,5,7,…

Repeat for d=3,5,7,…sqrt(n)

- if d is divisor then n is not prime.

Prime numbers (2)

Function to test something are named

isSomething()

and return either 0 or 1.

Important algorithms with multiple applications

Note: it only traverses the odd numbers up to sqrt(n)

```
def isPrime(n):

    if n==2 or n == 3 :
        return 1
    #endif

    if n % 2 == 0 :
        return 0
    #endif

    for d in range(3, math.sqrt(n)+1, 2)
            :
        if n % d == 0 :
            return 0
        # endif
    # endfor

    return 1

# end isPrime
```

Problems with Prime numbers

- Write a program to list all primes up to a threshold.
- Write a program to list all palyndrom primes.
- Write a program to list all primes whose mirror is prime too.
- Given an even number n then write a program to decompose n in a sum of 2 primes e.g. n = p1+p2 with p1 and p2 primes [Goldbach's conjecture]
- Etc.

```
def printPrimes():

    n = int(input('input n: '))

    # travers all numbers
    for i in range(2, n+1) :

        # test primality
        if isprime(i) :
            print(i)
        # endif
    # endfor
# end printPrimes
```

Find the reverse / mirror of a long.

If n is a long number then

last=n % 10 is the last digit.

n = n/10 is the new number from which we remove last.

n = n*10 +digit is the new number to which digit is added.

If we repeat removing the last digit then we can find:

- all the digits of the number.
- number of digits.
- the mirror of a number

We repeat while the number is not fully done.

```
def reverse(n):

    mirrorN = 0

    while n != 0 :
        digit = n % 10
        n = n // 10
        mirrorN = mirrorN * 10 + digit
    #endwhile

    return mirrorN
#end reverse
```

```
def isPalindrome(n):

    mirror = reverse(n)
    if n == mirror :
        return 1
    else :
        return 0
    #endif

#end isPalindrome
```

Greatest Common Divisor

If a, b are integer then the greatest common divisor

Example:

- a=24, b=36 ➔ gcd=12
- a=25, b=33 ➔ gcd=1

How to solve it?

- the Euclidian algorithm

The Euclidean Algorithm

Sequence of divisions until the remainder is 0.

Repeat the following:

- divide the numbers
- if remainder is 0 then break
- change the numbers:

first=second; second=remainder

```
def gcd(a,b):

    first, second = a, b

    while first % second != 0 :

        # int divide first to second
        remaider = first % second
        result   = first //second

        # prepare the next division
        first  = second
        second = remaider
    # endwhile

    return second

# end gcd
```

To do List

1. Read more about working with functions.
2. Solve the HW problems.