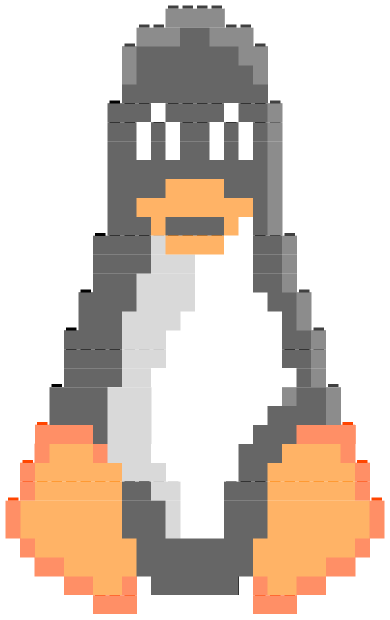
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΕΦΑΡΜΟΣΜΕΝΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΦΥΣΙΚΗΣ

ΦΥΣ 140 Εισαγωγή στην Επιστημονική Χρήση Υπολογιστών (15821)
Χειμερινό Εξάμηνο 2023

Φώτης Πτωχός και Αλέξανδρος Αττίκης

Φροντιστήριο 3

26 Σεπτεμβρίου 2023
15:00 - 17:00



Φροντιστήριο 3

Παράδειγμα 1 Παραδείγματα δομής επαναληπτικής διαδικασίας με τη χρήση βρόχου *for* (for loop) και βρόχου *while* (while loop):

tutorial3/ex1.py

```
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex1.py
   python3 ex1.py
   script -q ex1.log python3 -i ex1.py


DESCRIPTION:
Revisiting for/while xloops


LINKS:
https://www.w3schools.com/python/python_for_loops.asp
'''

print("\n=== Example of for-loop:")
for l in "KNOPFLER":
    print(l)


print("\n=== Another for-loop example:")
for x in range(0, 12, 2):
    print("\tx = %d" % (x) )
print("=== NOTE that the range(0, 12) is not the values of 0 to 12, but the
    values 0 to 12-step (\ie 0 to 10).")



print("\n=== Another for-loop example:")
for x in range(10, -2, -2):
    print("\tx = %d" % (x) )
print("=== NOTE that the range(10, -2, -2) is not the values of 10 to -2, but
    the values 10 to END-STEP: -2-(-2) = 0")


print("\n=== A simple while-loop example:")
i=-1
while i < 10:
    i+=1
    if i == 0:
        print("\ti = 0")
    elif i == 1:
        print("\ti = %d" % (i) )
    else:
        continue # can stop the current iteration of the loop, and continue
    with the next
```

```python


print("\n=== For-loop over list items:")
time = ["And then one day you find", "Ten years have got behind you", "No one told you when to run", "You missed the starting gun"]
for l in time:
    print("\t"+l)


print("\n=== While-loop over list items:")
echoes = ["The", "echo", "of", "a", "distant", "time", "comes", "willowing", "across", "the", "sand"]
a=0
while a < len(echoes):
    print(str(a) + ")", echoes[a])
    a+=1


print("\n=== Iterate over elements in an iterable (\eg list, tuple, or string) while keeping track of the index of the current element. It returns an iterator that produces pairs of index and value for each item in the iterable")
for a, echo in enumerate(echoes, start=0):
    print(str(a) + ")", echo, " = ", echoes[a])
    #print("%i) %s = %s" % (i, echo, echoes[i]))

print("\n=== Quit!")
quit()
```

Αποτέλεσμα:

```

=== Example of for-loop:
K
N
O
P
F
L
E
R

=== Another for-loop example:
        x = 0
        x = 2
        x = 4
        x = 6
        x = 8
        x = 10
```

```
=== NOTE that the range(0, 12) is not the values of 0 to 12, but the values
   0 to 12-step (\ie 0 to 10).

=== Another for-loop example:
        x = 10
        x = 8
        x = 6
        x = 4
        x = 2
        x = 0
=== NOTE that the range(10, -2, -2) is not the values of 10 to -2, but the
   values 10 to END-STEP: -2-(-2) = 0

=== A simple while-loop example:
        i = 0
        i = 1

=== For-loop over list items:
        And then one day you find
        Ten years have got behind you
        No one told you when to run
        You missed the starting gun

=== While-loop over list items:
0) The
1) echo
2) of
3) a
4) distant
5) time
6) comes
7) willowing
8) across
9) the
10) sand

=== Iterate over elements in an iterable (\eg list, tuple, or string) while
   keeping track of the index of the current element. It returns an iterator
    that produces pairs of index and value for each item in the iterable
0) The  =  The
1) echo  =  echo
2) of  =  of
3) a  =  a
4) distant  =  distant
5) time  =  time
6) comes  =  comes
7) willowing  =  willowing
8) across  =  across
9) the  =  the
10) sand  =  sand
```

```
=== Quit!
```

Στη δομή επανάληψης *for* το πλήθος των επαναλήψεων είναι δεδομένο και προκαθορισμένο, πριν αρχίσουν οι επαναλήψεις. Στη δομή επανάληψης *while* το πλήθος των επαναλήψεων καθορίζεται κατά τη διάρκεια της εκτέλεσης των εντολών εντός του βρόχου. Η αρχική τιμή της εν λόγω μεταβλητής καθορίζεται εκτός βρόχου, πριν το τμήμα του κώδικα δομής επανάληψης.

Παράδειγμα 2 Παραδείγματα για τη χρήση εμφωλευμένων βρόχων (nested loops):

tutorial3/ex2.py

```python
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex2.py
   python3 ex2.py
   script -q ex2.log python3 -i ex2.py


DESCRIPTION:
Introduction to nested loops


LINKS:
https://www.w3schools.com/python/python_for_loops.asp
'''
print("\n=== Nested for-loop example:")
for i in range(0, 3, 1):
    print("i = %d" % (i) )
    for j in range(0, 3, 1):
        print("\tj = %d" % (j) )


print("\n=== Another nested for-loop example:")
colour = ["red", "yellow", "green"]
fruit  = ["strawberry", "banana", "apple"]
for c in colour:
    for f in fruit:
        # The "inner loop" will be executed one time for each iteration of the
    "outer loop"
        print(c, f)


print("\n=== A double-nested for-loop example:")
# Outer loop from 2 to 3
for i in range(3, 0, -1):

    # Inner loop
    for c in "ABC": #"ROYGBIV":

        # Second inner loop
        for d in ["x", "y", "z"]: #["SATURDAY" , "SUNDAY"]:
            print("\t", i, c, d)
        print("")
print("=== Finished nested for-loops")


print("\n=== A nested while-loop:")
i = 3
while i > 0:

    j = 0
    while j < 4:
        print("\ti = %d, j = %d" % (i, j) )
        j+=1

    print("\t\ti = %d, j = %d" % (i, j) )
    i -= 1


print("\n=== A nested for-loop and while-loop example:")
num    = None
grades = []
names  = ["Costantinos", "Eleni", "Giorgos", "Myrto"]
for name in names:
    while num not in [i for i in range(0, 10, 1)]:
        try:
            num = int(input("Please type a grade for '" + name + "' in range
   [0, 9]: "))
        except:
            msg = "Invalid input '%s' of type %s (not type integer). Please try
    again" % (num, type(num))
            print(msg)
    grades.append(num)
    num = None

if len(grades) != len(names):
    print("List length mismatch! (%d != %d)" % (len(grades), len(names)))
else:
    print("\n=== Printing the grades of %d students: ", len(grades) )
    for i in range(0, len(grades), 1):
        print("\t", names[i], grades[i])

print("\n=== Quit!")
quit()
```

Αποτέλεσμα:

```
=== Nested for-loop example:
i = 0
  j = 0
  j = 1
  j = 2
i = 1
  j = 0
  j = 1
  j = 2
i = 2
  j = 0
  j = 1
  j = 2

=== Another nested for-loop example:
red strawberry
red banana
red apple
yellow strawberry
yellow banana
yellow apple
green strawberry
green banana
green apple

=== A double-nested for-loop example:
   3 A x
   3 A y
   3 A z

   3 B x
   3 B y
   3 B z

   3 C x
   3 C y
   3 C z

   2 A x
   2 A y
   2 A z

   2 B x
   2 B y
   2 B z

   2 C x
   2 C y
   2 C z

   1 A x
   1 A y
   1 A z

   1 B x
   1 B y
   1 B z

   1 C x
   1 C y
   1 C z

=== Finished nested for-loops
=== A nested while-loop:
  i = 3, j = 0
  i = 3, j = 1
  i = 3, j = 2
  i = 3, j = 3
    i = 3, j = 4
  i = 2, j = 0
  i = 2, j = 1
  i = 2, j = 2
  i = 2, j = 3
    i = 2, j = 4
  i = 1, j = 0
  i = 1, j = 1
  i = 1, j = 2
  i = 1, j = 3
    i = 1, j = 4

=== A nested for-loop and while-loop example:
Please type a grade for 'Costantinos' in range [0, 9]: 4
Please type a grade for 'Eleni' in range [0, 9]: 1
Please type a grade for 'Giorgos' in range [0, 9]: 9
Please type a grade for 'Myrto' in range [0, 9]: 8

=== Printing the grades of %d students:  4
   Costantinos 4
   Eleni 1
   Giorgos 9
   Myrto 8

=== Quit!
```

Σημαντικά σημεία που πρέπει να προσέχουμε:

- Ο εσωτερικός βρόχος πρέπει να βρίσκεται εξ' ολοκλήρου μέσα στον εξωτερικό βρόχο.
- Ο βρόχος που ξεκινάει τελευταίος ολοκληρώνεται πρώτος, ενώ ο βρόχος που ξεκινάει πρώτος τελειώνει τελευταίος.
- Η μεταβλητή / δείκτης δύο ή περισσότερων εμφωλευμένων βρόχων δεν μπορεί να έχουν την ίδια ονομασία!

Παράδειγμα 3 Παράδειγμα για τη χρήση συνάρτησης (*function*) σε συνδυασμό με τη δομή *try*/ *except*:

tutorial3/ex3.py

```
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex3.py
   python3 ex3.py
   script -q ex3.log python3 -i ex3.py


DESCRIPTION:
Introduction to functions


LINKS:
https://www.w3schools.com/python/python_operators.asp
https://www.tutorialspoint.com/python3/python_functions.htm
https://www.w3schools.com/python/trypython.asp?filename=demo_oper_mod
'''
def Print(msg, header=True):
    '''
    msg ....: string to be printed

    return..: nothing
    '''
    if (header):
        print("=== ex3.py:\n\t", msg)
    else:
        print("\t", msg)


def f(x):
    '''
    x .......: value at which the function f(x) must be evaluated at

    return ..: the value of my function at x
    '''
    return x+1

def g(x):
    '''
    x ......: value at which the function g(x) must be evaluated at

    return ..: the value of my function at x
    '''
    return x + 2

def h(x):
    '''
    x ......: value at which the function h(x) must be evaluated at

```

```python
    return ..: the value of my function at x
    '''
    return x + 3

try:
    msg = "=== ex3.py: Please type integer numbers in range [0, infty]: "
    x = int(input(msg))
except:
    Print("You did not type in an integer number in range [0, infty]? Quiting")
    quit()

Print('x = %s' % (x) )
Print('f(x) = %s' % f(x), False)
Print('g(x) = %s' % g(x), False)
Print('h(x) = %s' % h(x), False)
Print('g( h(x) ) ) = %s' % g( h(x)), False)
Print('f( g( h(x) ) ) = %s' % f( g( h(x) )), False)

Print("Quit!")
quit()
```

Αποτέλεσμα:

```
=== ex3.py: Please type integer numbers in range [0, infty]: 0
=== ex3.py:
        x = 0
        f(x) = 1
        g(x) = 2
        h(x) = 3
        g( h(x) ) ) = 5
        f( g( h(x) ) ) = 6
=== ex3.py:
        Quit!
```

Σημαντικά σημεία που πρέπει να κρατήσουμε όσο αφορά τις συναρτήσεις:

- Είναι επαναχρησιμοποιήσιμα μέρη προγραμμάτων (ή, αν προτιμάτε, μίνι προγράμματα εντός του κύριου προγράμματός σας).
- Αποτελούνται από ένα σύνολο εντολών που μπορούν να εκτελεστούν από οπουδήποτε στο πρόγραμμα και όσες φορές θέλουμε, διαδικασία που ονομάζεται κλήση της συνάρτησης (*calling*).
- Εκτελούμε μια συνάρτηση καλώντας το όνομα της και παρέχοντας τιμές για τις προκαθορισμένες παραμέτρους της. Οι παράμετροι είναι προαιρετικές, αλλά εάν η συνάρτησή σας δέχεται παραμέτρους τότε πρέπει να τις δηλώσετε μέσα στις παρενθέσεις.

Παράδειγμα 4 Άλλο ένα παράδειγμα χρήσης συνάρτησης σε συνδυασμό με *try/ except*:

tutorial3/ex4.py

```
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex4.py
   python3 ex4.py
   script -q ex4.log python3 -i ex4.py


DESCRIPTION:
Introduction to functions


LINKS:
https://www.w3schools.com/python/python_for_loops.asp
'''
def CelciusToFarenheit(C):
    '''
    C ....: temperature in degrees Celcius

    return: temperature in degrees Farenheit
    '''
    F = (C*9/5) + 32
    return F

# Declare empty lists
Celcius=[]
print("=== Please provide 10 temperature values in degrees Celcius: ")
counter = 1
while len(Celcius) < 10:
    try:
        Celcius.append(float(input("\t%d) " % (counter))) )
        counter+=1
    except:
        print("\tInvalid temperature value. Please try again!")


Farenheit = [CelciusToFarenheit(C) for C in Celcius]

print("\n=== Conversion table:\n {:>8} {:>8}".format("(C)", "(F)"))
for i in range(0, len(Celcius), 1):
    # These curly braces "{}" serve as placeholders for values that will be
   inserted into the string
    # The colon ":" indicates the start of the format specification
    # The 8 is the total width of thae field. It means that the total width for
    the printed number, including both digits and spaces, will be 8 characters.
    # The ".1" specifies the number of decimal places to display. In this case,
    it's set to 1 decimal place.
    # The "f" indicates that the value to be inserted is a floating-point
   number
    print('{:8.1f} {:8.1f}'.format(Celcius[i], Farenheit[i]))
```

```python



print("\n=== Quit!")
quit()
```

Αποτέλεσμα:

```
=== Please provide 10 temperature values in degrees Celcius:
1) 0
2) 10
3) 20
4) 30
5) 40
6) 50
7) 60
8) 70
9) 80
10) 100

=== Conversion table:
      (C)       (F)
     0.0      32.0
    10.0      50.0
    20.0      68.0
    30.0      86.0
    40.0     104.0
    50.0     122.0
    60.0     140.0
    70.0     158.0
    80.0     176.0
   100.0     212.0

=== Quit!
```

Θα εξετάσουμε λεπτομερώς τη μέθοδο *format* στις επόμενες διαλέξεις.

Παράδειγμα 5 Παράδειγμα εκτέλεσης συνάρτησης εντός μιας άλλης συνάρτησης:

tutorial3/ex5.py

```python
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex5.py
   python3 ex5.py
   script -q ex5.log python3 -i ex5.py


DESCRIPTION:
Introduction to functions


LINKS:
https://www.w3schools.com/python/python_operators.asp
https://www.tutorialspoint.com/python3/python_functions.htm
https://www.w3schools.com/python/trypython.asp?filename=demo_oper_mod
'''
def Print(msg, header=True):
    '''
    msg ....: string to be printed

    return..: nothing
    '''
    if (header):
        print("=== example5.py:\n\t", msg)
    else:
        print("\t", msg)

def isVowel(c):
    '''
    check if c, type string, is a vowel or not

    c : input character (string)

    return: true if vowel, false if consonant
    '''
    letter = c.lower()
    vowels = ["a", "e", "i", "o", "u"]

    # Return True if c is a vowel
    if letter in vowels:
        return True
    else:
        return False

def getVowelsFromPhrase(phrase):
    '''
     Takes a phrase, and returns a string of all the vowels

     Initalize an empty string to hold all of the vowels

    phase: string you want to loook for vowels in
    return list of vowels (string) found in phrase
    '''
    vowel_string = ''
    vowels = []
    for letter in phrase:
        # check if each letter is a vowel
        if isVowel(letter):
            vowels.append(letter)
        else:
            # if not a vowel, we don't care about it- so do nothing!
            pass #continue

    return vowels

myPhrase = ""
try:
    msg = "Please type a phrase to look for vowels: "
    myPhrase = input(msg)
except:
    msg = "Something went wrong. Please type a phrase"
    Print(msg)

vList = getVowelsFromPhrase(myPhrase)
Print("Found %d vowels in the phrase '%s':" % (len(vList), myPhrase) )
for i,v in enumerate(vList, 1):
    Print("%d) %s" % (i, v), i==0)

quit()
```

Αποτέλεσμα:

```
Please type a phrase to look for vowels: abcdefghijklmnopqrstuvwxyz
=== example5.py:
         Found 5 vowels in the phrase 'abcdefghijklmnopqrstuvwxyz':
         1) a
         2) e
         3) i
         4) o
         5) u
```

Παράδειγμα 6 Πρόγραμμα που δημιουργεί μια λίστα με όλους τους πρώτους αριθμούς σε ένα εύρος ακεραίων αριθμών που καθορίζει ο χρήστης, όπως το παράδειγμα από τη Διάλεξη 03:

tutorial3/ex6.py

```python
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex6.py
   python3 ex6.py
   script -q ex6.log python3 -i ex6.py


DESCRIPTION:
Introduction to functions


LINKS:
https://www.w3schools.com/python/python_operators.asp
https://www.tutorialspoint.com/python3/python_functions.htm
https://www.w3schools.com/python/trypython.asp?filename=demo_oper_mod
'''
def Print(msg, header=True):
    '''
    msg ....: string to be printed

    return..: nothing
    '''
    if (header):
        print("===", __file__, "\n\t" + msg)
    else:
        print("\t", msg)


def GetLargestPrime(minNum, maxNum):
    '''
    returns largest prime number in range [minNum, maxNum]

    minNum: smallest integer to use in search range
    maxNum: largest integer to use in search range

    return: integer
    '''
    if (minNum < 2 or maxNum < 2):
        msg = "Invalid search range [%d, %d]. The smallest integer is 2!" % (
    minNum, maxNum)
        raise Exception(msg)


    if (minNum > maxNum):
        msg = "The maximum integer (=%d) must have a larger value than the
    minimum integer (=%d)" (maxNum, minNum)
        raise Exception(msg)

    # Declare variables
    myPrimes = []
    i = minNum #2

    # Outside while-loop
    while(i <= maxNum):
        j = minNum

        # Nested while-loop (only need to consider numbers halfway through the range)
        while(j <= (i/j)):

            # Use modulo arithmetic operator to see if i is perfectly divisible by j
            if not(i % j):
                # division remainder is 0 => i divisible by j => not a prime!
                break
            # Increment index of nested-loop  (until you find a prime number)
            j = j + 1

        # Check if j is a prime. If yes append to the list of primes
        if (j > i/j):
            if 0:
                Print("%d is prime!" % (i))
            myPrimes.append(i)

        # Increment index for outside loop
        i = i + 1

    Print("List of prime numbers found: %s" % (myPrimes))
    return myPrimes[-1]



print("\n=== Please type two integer numbers, i and j, so that i < j and i >1: ")
try:
    minNum = (int(input("\tMinimum: ")))
    maxNum = (int(input("\tMaximum: ")))
    Print("Largest prime in range [%d, %d] is %d" % (minNum, maxNum, GetLargestPrime(minNum, maxNum)))
except:
    Print("Something went wrong. Please see below the docstrings of the %s function!" % (GetLargestPrime.__name__))
    Print(GetLargestPrime.__doc__)
    help(GetLargestPrime)

Print("Quit!")
quit()
```

Αποτέλεσμα:

```
=== Please type two integer numbers, i and j, so that i < j and i >1:
Minimum: 2
Maximum: 100
=== ex6.py
        List of prime numbers found: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29,
   31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
=== ex6.py
        Largest prime in range [2, 100] is 97
=== ex6.py
        Quit!
```

Παράδειγμα 7 Πρόγραμμα που κάνει χρήση συνάρτησης χρήστη για τον υπολογισμό του παραγοντικού ενός οποιοδήποτε αριθμού που περνάει ως όρισμα:

tutorial3/ex7.py

```python
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex7.py
   python3 ex7.py
   script -q ex7.log python3 -i ex7.py


DESCRIPTION:
Example of a function that calls itself! This program consists of several
    functions and an
interactive section that calculates the factorial of a given positive integer.
The function factorial() calculates the factorial of a positive integer x using
     recursion.
It includes input validation to ensure that x is a non-negative integer.
If x is negative, it raises an exception with an error message. if x is 0 or 1,
     it returns 1,
as the factorial of 0 and 1 is defined to be 1. For values of x greater than 1,
     it calculates
the factorial by calling itself recursively with x-1 and multiplying the result
     by x.


The program takes user input to enter a positive integer x for which the
    factorial will be calculated.
It calls the factorial(x) function and prints the result along with a header if
     everything goes well.
If an exception occurs (e.g., invalid input or an error in the factorial
    function), it prints an error
 message and displays the docstring of the factorial function.
The program prints "Quit!" before exiting.


LINKS:
https://www.w3schools.com/python/python_operators.asp
https://www.tutorialspoint.com/python3/python_functions.htm
https://www.w3schools.com/python/trypython.asp?filename=demo_oper_mod
'''
def Print(msg, header=True):
    '''
    msg ....: string to be printed

    return..: nothing
    '''
    if (header):
        print("===", __file__, "\n\t" + msg)
    else:
        print("\t", msg)

```

```python
def factorial(x):
    '''
    returns the factorial of positive integer number  x

    x: positive integer number (>0)

    return: integer
    '''
    if (x < 0):
        msg = "Cannot evaluate factorial of negative numbers (undefined)"
        raise Exception(msg)

    if (x in [0, 1]):
        return 1
    else:
        return x*factorial(x-1) # function calls itself

def factorialAlt(x):
    '''
    As shown in Lecture 03, page 13
    '''
    f = 1.0
    for k in range(1, n+1):
        f*=k
    return f


try:
    msg = "=== Please type integer numbers in range [0, infty]: "
    x = int(input(msg))
    Print("factorial(%d) = %d" % (x, factorial(x)))
except:
    Print("Something went wrong. Please see below the docstrings of the %s
   function!" % (factorial.__name__))
    Print(factorial.__doc__)
    #help(factorial)

Print("Quit!")
quit()
```

Αποτέλεσμα:

```
=== Please type integer numbers in range [0, infty]: 5
=== ex7.py
        factorial(5) = 120
=== ex7.py
        Quit!
```

Το πρόγραμμα ξεκινά με μια σειρά σχολίων (*docstring*) που περικλείεται σε τριπλά εισαγωγικά (”’) και παρέχει πληροφορίες σχετικά με το τι κάνει το πρόγραμμα και οι συναρτήσεις του. Ακολουθεί ο ορισμός συναρτήσεων υπολογισμού παραγοντικού factorial() και factorialAlt() που παίρνουν ένα μόνο όρισμα x . Το πρόγραμμα διασφαλίζει ότι το x είναι ένας θετικός ακέραιος αριθμός, αφού τα παραγοντικά αρνητικών ακεραίων δεν έχουν καθορισμένη σημασία. Εάν το x είναι αρνητικό, δημιουργεί μια εξαίρεση με ένα μήνυμα σφάλματος που υποδεικνύει ότι το παραγοντικό δεν έχει οριστεί για αρνητικούς αριθμούς. Επίσης, το πρόγραμμα ελέγχει για δύο βασικές περιπτώσεις· Εάν το x είναι 0 ή 1, επιστρέφει 1. Αυτό συμβαίνει επειδή το παραγοντικό του 0 και του 1 ορίζεται ως 1. Σε περίπτωση που το x είναι μεγαλύτερο από 1, το πρόγραμμα υπολογίζει το παραγοντικό πολλαπλασιάζοντας το x με το παραγοντικό του x-1. Αυτό γίνεται καλώντας τη συνάρτηση factorial() μέσα της με το όρισμα x-1. Αυτή η αναδρομική διαδικασία συνεχίζεται μέχρι να φτάσει σε μία από τις βασικές περιπτώσεις, οπότε η συνάρτηση επιστρέφει τη τελική τιμή του υπολογισμού.