**BÀI TẬP THỰC HÀNH SỐ 1**

**Bài 1**: Hoàn thành 25/150 bài tập cơ bản ở w3resourse.com

1. **Write a Python program to print a string with specific format**

print("Twinkle, twinkle, little star,\n\tHow I wonder what you are!\n\t\tUp above the world so high,\n\t\tLike a diamond in the sky.\nTwinkle, twinkle, little star,\n\tHow I wonder what you are")

1. **Write a Python program to get the Python version you are using**

*# Method 1*

**from** os **import** system

system("py --version")

*# Method 2*

**import** sys

print(sys.version)

*# Method 3*

**import** platform

print(platform.python\_version())

1. **Write a Python program to display the current date and time**

**import** datetime

print(datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"))

1. **Write a Python program which accepts the radius of a circle from the user and compute the area**

**import** math

**try**:

    radius **=** float(input("Enter the radius of the circle: "))

**except**:

    print("Error. The character you type seem not a number. Please try again")

**def** FindCircleArea(**radius**): **return** radius**\*\***2**\***math.pi

print(FindCircleArea(radius))

1. **Write a Python program which accepts the user’s first and last name and print them in reverse order with a space between them**

first\_name = input("Enter your first name: ")

last\_name = input("Enter your last name: ")

print(last\_name + " " + first\_name)

1. **Write a Python program wich accepts a sequence of comma-serparated numbers from user and generate a list and a tuple with those number**

**Sample data: 3,5,7,23**

data **=** "3,5,7,23"

list\_data **=** data.split(",")

tuple\_data **=** tuple(list\_data)

print(list\_data)

print(tuple\_data)

1. **Write a Python program to accept a filename from the user and print the extension of that**

filename **=** input("Enter your filename: ")

filename\_analyse **=** filename.split('.')

print("The extension of this file is: "**+**filename\_analyse[**-**1])

1. **Write a Python program to display the first and the last colors from the following list**

color\_list = ["Red","Green","White" ,"Black"]

color\_list **=** ["Red", "Green", "White", "Black"]

print("First color: "**+**color\_list[0])

print("Last color: "**+**color\_list[**-**1])

1. **Write a Python program to display the examination schedure.**

exam\_st\_date **=** (11, 12, 2014)

print("The examination will start from: %i/ %i /%i" **%** exam\_st\_date)

1. **Write a Python program that accepts an interger () and computes the value of n+nn+nnn**

number\_a **=** int(input("Enter a number: "))

*# Method 1*

number\_aa **=** int("%s%s" **%** (number\_a, number\_a))

number\_aaa **=** int("%s%s%s" **%** (number\_a, number\_a, number\_a))

*# Method 2*

number\_aa **=** 10**\***number\_a**+**number\_a

number\_aaa **=** 100**\***number\_a**+**number\_aa

print(number\_a**+**number\_aa**+**number\_aaa)

1. **Write a Python program to print the documents (syntax, decription, etc) of Python buil-in function**

print(abs.\_\_doc\_\_)

1. **Write a Pythom program to print the calendar of a given month and year**

**import** calendar

month**=**int(input("Enter a month: "))

year**=** int(input("Enter a year: "))

print(calendar.month(year, month))

1. **Write a Python program to print the following document**

print("""

a string that you "don't" have to escape

This

is a ....... multi-line

heredoc string --------> example

""")

1. **Write a Python program to calculate number of days between two date**

**from** datetime **import** date

first\_date **=** date(2014, 7, 2)

last\_date **=** date(2014, 7, 11)

result **=** last\_date**-**first\_date

print(result.days)

1. **Write a Python program to get the volume of a sphere with radius 6**

**import** math

FindVolume**=**lambda **radius**: (4**/**3)**\***math.pi**\***radius**\*\***3

print(FindVolume(6))

1. **Write a Python program to get the difference between a given number and 17, if the number is greater than 17 returen double the absolute difference**

number **=** int(input("Enter a number: "))

result **=** number **-** 17

**if** result **>** 0:

    result **\*=** 2

**elif** result **<** 0:

    result**=**abs(result)

print(result)

1. **Write a Python program to test whether a number is within 100 of 1000 or 2000**

**def** CheckWithin(**number**):

**return** (abs(1000**-**number) **<=** 100) **or** (abs(2000**-**number) **<=** 100)

number **=** int(input("Enter a number: "))

print(CheckWithin(number))

1. **Write a Python program to calculate the sum of three given numbers, if the values are equal then return three times of their sum**

**def** GetSum(**a**, **b**, **c**):

    result **=** a**+**b**+**c

**if** a **==** b **and** a **==** c:

**return** result**\***3

**return** result

a **=** int(input("Enter first number: "))

b **=** int(input("Enter second number: "))

c **=** int(input("Enter third number: "))

print(GetSum(a, b, c))

1. **Write a Python program to get a new string from a givenb string where “Is” has been added to the front. If the given string already begin with “Is” then return unchanged**

**def** AddIs(**line**):

**if** len(line) **<** 2 **or** line[:2] **!=** "Is":

**return** "Is"**+**line

**return** line

line **=** input("Enter a string: ")

print(AddIs(line))

1. **Write a Python prgram to get a string which is n (non-negative integer) copies of a given string**

**def** CoppyString(**line**, **num\_of\_copy**):

    result **=** ""

**for** i **in** range(num\_of\_copy):

        result **+=** line

**return** result

line **=** input("Enter a string to copy: ")

num\_of\_copy **=** int(input("Enter a number of copies: "))

**if** num\_of\_copy **>=** 0:

    print(CoppyString(line, num\_of\_copy))

**else**:

    print("Invalid number of copies")

1. **Write a Python program to find whether a given number (accept from the user) is even or odd, print out appropiate message to the user**

**def** CheckEven(**number**): **return** number **&** 1 **==** 0

**try**:

    number **=** int(input("Enter an integer number: "))

**if**(CheckEven(number)):

        print("{} is an even number!".format(number))

**else**:

        print("{} is an odd number!".format(number))

**except**:

    print("Invalid number")

1. **Write a Python program to count the number 4 in a given list**

**from** numpy.random **import** randint

**def** CountFour(**arr**):

    count **=** 0

**for** number **in** arr:

**if** number **==** 4:

            count **+=** 1

**return** count

length **=** int(input("Enter the length of the number list: "))

number\_list **=** randint(10, **size=**(length))

print("The number list used for counting: {}".format(number\_list))

print("The number of number '4' in the given list is: {}".format(CountFour(number\_list)))

1. **Write a Python program to get the n (non-negative number) copies of the first 2 characters of a given string. Return the n copies of the whole string if the length is less than 2**

**def** Copy(**str**, **num\_of\_copy**):

    result **=** ""

**for** i **in** range(num\_of\_copy):

        result **+=** *str*

**return** result

**def** Copy\_2FirstCharacters(**str**, **num\_of\_copy**):

**if** len(str) **<** 2:

**return** Copy(str, num\_of\_copy)

**else**:

**return** Copy(str[:2], num\_of\_copy)

*str* **=** input("Enter a string: ")

num\_of\_copy **=** int(input("Enter the number of copies: "))

print("The result of the copy is: "**+**Copy\_2FirstCharacters(str, num\_of\_copy))

1. **Write a Python program to test whether a passed letter is a vowel or not**

**def** CheckVowel(**letter**):

    vowel\_list **=** ['a', 'e', 'i', 'o', 'u']

**for** character **in** vowel\_list:

**if** letter.lower() **==** character:

**return** True

**return** False

letter **=** input("Enter a letter: ")

**if** len(letter)**>**1:

    print("Invalid letter!")

**elif** CheckVowel(letter):

    print(letter**+**" is a vowel")

**else**:

    print(letter**+**" is a consonant")

1. **Write a Python program to check whether a specified value is contained in a group of values**

**from** numpy.random **import** randint

length **=** int(input("Enter the length of the number list: "))

number\_list **=** randint(10, **size=**length)

print("The number list used for checking: {}".format(number\_list))

check\_number**=**int(input("Enter the number to check: "))

**if** number\_list.\_\_contains\_\_(check\_number):

    print("{} is contained in the list".format(check\_number))

**else**:

    print("{} is not contained in the list".format(check\_number))

**from** numpy.random **import** randint

**def** BinarySearch\_Recursion(**arr**, **number**, **left\_index**, **right\_index**):

**if** left\_index **>** right\_index:

**return** False

    mid\_index **=** int((left\_index**+**right\_index)**/**2)

**if** arr[mid\_index] **==** number:

**return** True

**if** number **<** arr[mid\_index]:

**return** BinarySearch\_Recursion(arr, number, left\_index, mid\_index**-**1)

**else**:

**return** BinarySearch\_Recursion(arr, number, mid\_index**+**1, right\_index)

**def** BinarySearch(**arr**, **number**):

**return** BinarySearch\_Recursion(arr, number, 0, len(arr)**-**1)

length **=** int(input("Enter the length of the number list: "))

number\_list **=** randint(100, **size=**length)

number\_list.sort()

print("The number list used for checking: {}".format(number\_list))

check\_number **=** int(input("Enter the number to check: "))

**if** BinarySearch(number\_list, check\_number):

    print("{} is contained in the list".format(check\_number))

**else**:

    print("{} is not contained in the list".format(check\_number))

**Bài 2**: Viết hàm thực hiện các chức năng

1. Tính:
2. a+b

Sum **=** lambda **a**, **b** : a **+** b

1. a/b

Divide **=** lambda **a**, **b** : a **/** b

1. ab

Exponential **=** lambda **a**, **b** : a **\*\*** b

1. Tính diện tích hình chữ nhật khi biết bán kính ??? Tính diện tích hình tròn?

**import** math

GetArea **=** lambda **radius** : radius **\*\*** 2 **\*** math.pi

1. Xuất tất cả các số nguyên tố trong 1 khoảng cho trước

**def** IsPrime(**number**):

**if** number **<** 2:

**return** False

**for** i **in** range(2, number):

**if** number **%** i **==** 0:

**return** False

**return** True

**def** WritePrime(**first**, **last**):

    count **=** 0

**for** number **in** range(first**+**1, last):

**if** IsPrime(number):

            print(number, **end=**"\t")

            count **+=** 1

**if** count **%** 5 **==** 0:

                print("\n")

1. Kiểm tra 1 số nguyên n có phải là số Fibonacci hay không

**import** math

**def** IsPerfectSquare(**number**):

    temp **=** int(math.sqrt(number))

**return** temp**\*\***2 **==** number

**def** IsFibonacci(**number**):

**return** IsPerfectSquare(5**\***number**\*\***2**+**4) **or** IsPerfectSquare(5**\***number**\*\***2**-**4)

1. Tìm số Fibonacci thứ n (dùng đệ quy và không đệ quy)

**def** FindFibonacci\_Recursion(**number**):

**if** number **==** 0:

**return** 0

**if** number **==** 1:

**return** 1

**return** FindFibonacci\_Recursion(number**-**1)**+**FindFibonacci\_Recursion(number**-**2)

**def** FindFibonacci(**number**):

    number\_0 **=** 0

    number\_1 **=** 1

    number\_n **=** 1

**if** number **<** 0:

**return** **-**1

**if** number **==** 0 **or** number **==** 1:

**return** number

**for** i **in** range(2, number):

        number\_0 **=** number\_1

        number\_1 **=** number\_n

        number\_n **=** number\_0**+**number\_1

**return** number\_n

1. Tính tổng n số Fibonacci đầu tiên (dùng đệ quy và không dùng đệ quy)

**def** FindFibonacci\_Recursion(**number**):

**if** number **==** 0:

**return** 0

**if** number **==** 1:

**return** 1

**return** FindFibonacci\_Recursion(number**-**1)**+**FindFibonacci\_Recursion(number**-**2)

**def** SumFibonacci(**number**):

    sum **=** 0

**for** i **in** range(number):

        sum**+=**FindFibonacci\_Recursion(i)

**return** sum

**def** FindFibonacci\_Recursion(**number**):

**if** number **==** 0:

**return** 0

**if** number **==** 1:

**return** 1

**return** FindFibonacci\_Recursion(number**-**1)**+**FindFibonacci\_Recursion(number**-**2)

**def** SumFibonacci(**number**):

**if** number **==** 0:

**return** FindFibonacci\_Recursion(number)

**return** FindFibonacci\_Recursion(number)**+**SumFibonacci(number **-** 1)

1. Tính tổng căn bậc 2 của n số nguyên đầu tiên

**def** SumSquareRoot(**number**):

**if** number **==** 1:

**return** number

**return** math.sqrt(number)**+**SumSquareRoot(number **-** 1)

1. Giải phương trình bậc 2 ax2+bx+c=0

**import** math

**def** GetDelta(**a**, **b**, **c**):

**return** b**\*\***2**-**4**\***a**\***c

**def** SolveQuadraticEquation(**a**, **b**, **c**):

    roots **=** []

**if** GetDelta(a, b, c) **<** 0:

**return** roots

**if** GetDelta(a, b, c) **==** 0:

        roots.append(**-**b**/**2**\***a)

**else**:

        roots.append((**-**b**+**math.sqrt(GetDelta(a, b, c)))**/**(2**\***a))

        roots.append((**-**b**-**math.sqrt(GetDelta(a, b, c)))**/**(2**\***a))

**return** roots

1. Tính n!

**def** GetFactorial(**number**):

**if** number **==** 1:

**return** number

**return** number **\*** GetFactorial(number **-** 1)

1. In \* Dạng tam giác

**def** PrintTriangle(**edge**):

**for** i **in** range(1, edge **+** 1):

**for** j **in** range(1, i **+** 1):

            print("\*", **end=**" ")

        print()

1. Đổi giờ, phút, giây: Thời gian đầu vào là giây được đổi thành giờ, phút, giây,

**def** SwitchTime(**time**):

    temp **=** int(time)

    hour**=** int(temp**/**3600)

    temp**%=**3600

    minute**=**int(temp**/**60)

    second**=**temp**%**60

    print("{0}:{1}:{2}".format(hour,minute,second))

1. Cho 1 mảng số nguyên
2. Xuất tất cả các số lẻ không chia hết cho 5

**def** GetOddAndNotDivisibleBy5(**arr**):

    result **=** []

**for** number **in** arr:

**if** number **%** 2 **!=** 0 **and** number **%** 5 **!=** 0:

            result.append(number)

**return** result

1. Xuất tất cả các số Fibonacci

**import** math

**def** IsPerfectSquare(**number**):

    temp **=** int(math.sqrt(number))

**return** temp**\*\***2 **==** number

**def** IsFibonacci(**number**):

**return** IsPerfectSquare(5**\***number**\*\***2**+**4) **or** IsPerfectSquare(5**\***number**\*\***2**-**4)

**def** GetFibonacci(**arr**):

    result**=**[]

**for** number **in** arr:

**if** IsFibonacci(number):

            result.append(number)

**return** result

1. Tìm số nguyên tố lớn nhất

**def** IsPrime(**number**):

**if** number **<** 2:

**return** False

**for** i **in** range(2, number):

**if** number **%** i **==** 0:

**return** False

**return** True

**def** GetMaxPrime(**arr**):

    max **=** 0

**for** number **in** arr:

**if** IsPrime(number) **and** max **<** number:

            max **=** number

**return** max

1. Tìm số Fibonacci bé nhất

**import** math

**def** IsPerfectSquare(**number**):

    temp **=** int(math.sqrt(number))

**return** temp**\*\***2 **==** number

**def** IsFibonacci(**number**):

**return** IsPerfectSquare(5**\***number**\*\***2**+**4) **or** IsPerfectSquare(5**\***number**\*\***2**-**4)

**def** GetMinFibonacci(**arr**):

    min**=**arr[0]

**for** number **in** arr:

**if** IsFibonacci(number) **and** min**>**number:

            min**=**number

**return** min

1. Tính trung bình các số lẻ

**def** GetAverageOfOdd(**arr**):

    result**=**[]

**for** number **in** arr:

**if** number **&**1 **!=**0:

            result.append(number)

    average**=**0

**for** number **in** result:

        average**+=**number

**return** average**/**len(result)

1. Tính tích các phần tử là số lẻ không chia hết cho 3 trong mảng

**def** GetOddVolume(**arr**):

    result **=** 1

**for** number **in** arr:

**if** number **&** 1 **!=** 0 **and** number **%** 3 **!=** 0:

            result **\*=** number

**return** result

1. Đổi chỗ 2 phần tử trong danh sách

**def** Swap(**arr**, **first\_index**, **second\_index**):

    arr[first\_index],arr[second\_index]**=**arr[second\_index],arr[first\_index]

1. Đảo ngược trật tự các phần tử của danh sách

**def** Reverse(**arr**):

    arr.reverse()

k) Đếm số lần xuất hiện của một số trong danh sách

**def** CountElement(**arr**, **number**):

**return** arr.count(number)

l) Xuất các số xuất hiện n lần trong danh sách:

**def** CountElement(**arr**, **number**):

**return** arr.count(number)

**def** GetElementThatAppear(**arr**, **number\_of\_appearance**):

    result **=** []

**for** element **in** arr:

**if** CountElement(arr, element) **==** number\_of\_appearance **and** element **not** **in** result:

            result.append(element)

**return** result

m) Xuất các só xuất hiện nhiều lần trong danh sách

**def** CountElement(**arr**, **number**):

**return** arr.count(number)

**def** GetElementThatAppearMoreThanOnce(**arr**):

    result **=** []

**for** element **in** arr:

**if** CountElement(arr, element) **>=** 2 **and** element **not** **in** result:

            result.append(element)

**return** result