කැමති විෂය
කැමති වෙලාවක
කැමති තැනක
නිදහසේ ඉගන ගන්න
පාඩම් සහ ප්‍රශ්න

Shilpa64.lk

ශිල්පසයුර Google CSSL COMPUTER SOCIETY OF SRI LANKA

AlgoHack aims to teach Computer Science and Programing to young people, initiated by Shilpa Sayura Foundation, supported by GOOGLE RISE and Computer Society of Sri Lanka.

AlgoHack #6

PROGRAMING FUNCTIONS

AlgoHack #6

PROGRAMING FUNCTIONS

Authors
Niranjan Meegammana
N P Vishwa Kumara

Reviewers
Ravindu Ramesh Perera, Devanjith De Silva, Prabhashana Hasthidhara, Yamuna Ratnayake.

ශිල්පසයුර Google CSSL COMPUTER SOCIETY OF SRI LANKA

AlgoHack aims to teach Computer Science and Programing to young people, initiated by Shilpa Sayura Foundation, supported by GOOGLE RISE and Computer Society of Sri Lanka..

Functions are very interesting method in programing.
Think of your bicycle.

It has many functions.
The rotating the back wheel pushes the bike forward.
How do you rotate back wheel?
You have to push the paddle down
Paddle turns the sprocket,
Sprocket pulls the chain,
the chain rotates cog wheel fixed to the back wheel.
Every Time we push the paddle down the bike moves.

What are the other functions in your bicycle that help you ride safe, fast and joyfully.

A Prime Number Checking Program

```
i = 2
def isPrime(N):
    global i
    print ( "Processing ", N , "/", i)
    if (N % i == 0 and N > i ):# division
        return False
    i = i + 1
    if(N > i): # upto N-1
        prime=isPrime(N)
        return prime
    else:
        return True
N = int(input('Enter Number: '))
prime= isPrime(N)
if (prime):
    print ( N , " is prime")
else:
    print ( N , " is not prime")
```

Draw a flowchart for this program
Input numbers 1,8, 29, 35, 121, 243, 444, -1, 0
Is there a way to improve Prime program**?**
Think about maxDivisor=round ((N-1) / 2) + 1
Tip : declare max divisor as a global variable

Modify and improve isPrime2 (). Compare Solutions.

```
def isPrime2(n):
    for i in range(2, n):
        if (n%i == 0):
           return False
    return True
```

Prime Numbers are odd numbers bigger than 2.
Not all odd numbers are Prime numbers.
If we process all odd numbers > 2 then we can find prime numbers.
The method we use is **dividing recursively**
Lets pickup first number and name it **N**
We first divide **N** by 2
If the result is a whole number,
N is even number, therefore not prime
If the result has a decimal part
N can be a prime, we have to divide again
We now divide by 3
If the result has a decimal part,
we keep doing division

Should we divide **N** by 4 ?
Primes are odd numbers, can't divided by even numbers.

So we continue dividing by 3, 5, 7, 9 etc

Where do we stop dividing?
If we want to divide number **N** by number **n**,
then **n** has to be smaller than **N**.
So, we have to stop **n** at **N -1**.

Look at math division again.
After halfway of our recursive division process
We start getting results like 1 point something.
Example : 11/7=1.5 or 31/19=1.63
They will not divide the number in full.
Then we stop and say say **N is a Prime.**

Can we can find halfway number with **N/2** ?
Is this function **n=round((N-1)/2)+1** good for that?

Can you describe one function that you find interesting in your bicycle? Draw a flowchart.
Mine is the gear mechanism.

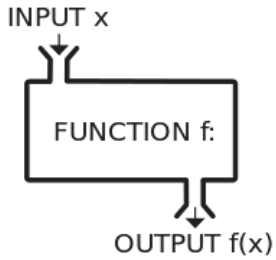
School Uniform Production

Let us assume that we have a computer making school uniforms. All uniforms are made in one design, but the sizes of uniforms can vary by student..
School uniform **process** have 3 **functions** for design, cutting and sawing. Program **inputs** are height, bust, waist, neck and sleeve length. **Output** is a dress.

Making a cup of tea. The inputs are amount of tea, milk and sugar. Different **inputs** give different taste. Process is making the tea in steps, **Output** is a cup of tea.

Create a flowchart for a smart tea machine.

Functions are designed to do specific things with code.



Functions take inputs.
Functions process inputs.
Functions return an output.
This program adds two numbers

```
a=2
b=3
c=a+b
print(c)
```

We can write this program **using a function**.
First we give a **meaningful name** to the function, then we describe what the function will do.
addFunction() : Takes two inputs and output the sum.

```
function addFunction(a,b) :
    c=a+b
    return c
#main program calls the function
result=addFunction(4,5)
Print (result)
```


Prime Number

A prime number is whole number greater than 1 that can only be divided by 1 and itself.
Examples : **2, 3, 5, 7, 11** ……. *more*

Primes numbers are good for **ordering** of non repeating values. Prime numbers play a central role in **number theory**, cryptography and computer security.

Professor CROW, asks his daughter who is good with division to **find Prime** Numbers from 20 Numbers. How can she find prime numbers fast?

Let us study this problem together.
We will follow **problem solving steps**,
first do it by hand,
learn, draw a flowchart and write a program.
Let's break the problem into steps.

Math Rules Tell Us
Prime number is > 1
Only 1 and itself can divide it.
Wow! All other numbers can't divide it.
We can ask more questions.
Is it a odd or even number?
2 is a prime number,
All other even numbers are divided by 2
So even numbers other than 2 not Primes.
We **Remove all even numbers > 2**
Our problem is getting solved now!
Summarize What we learned.

A **local variable** is **declared inside** a function. Only the **code inside** the function **can access** and modify it.

What happens in following code?

```
def func():
    x= "local variable"
    print("x inside function")
    print(x)

func()
print("outside function")
#print(x)
```

Remove the # comment and run.
What happens? Why?

Local variables are **destroyed** when function **exits**.
Code outside the function **cannot access** them.

What happens in following code?

```
x = 'global value'

def func():
    #global x
    x = 'local value'
    print('* inside function')
    print(x)

print('* before function')
print(x)
func()
print('* after function')
print(x)
```

Remove the # comment and run
What happens ? Why?

How functions work?

When we call a function, we transfers the program control to the function. The function **takes inputs**, executes its code to **process them** and **return an output** to the calling program. Calling program receives the output and executes next instruction after function.

Explain how a function works to a friend giving a real life example.

Functions are two types.
Programing languages comes with built in functions. The programmer can create own function using built-in functions. They are user defined functions.

Built-in functions in python
abs() function returns absolute value of a number

```
c= abs(2.2)
print (c) # result will be 2
```

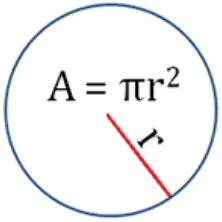
Some built-in functions in python
input() Reads a line from input.
int() Convert a string to an integer.
max() Return the largest item in a list
min() Return the smallest item in a list
str() Return a str version of a variable
sum() Sums an items list

User defined functions in python

```
function sum(a, b, c) :
    c= a+b+c
    return c

result=sum(2,3,1)
print(result) # 5 will be printed
```

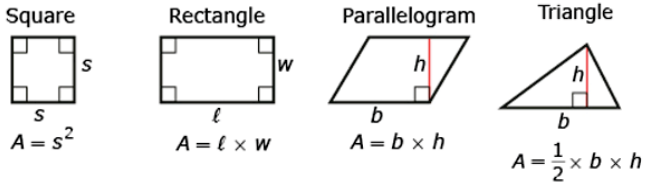
Program to calculate a area of a circle.




```
R=int(input())
Area=circleArea(radius)
print (Area)

def circleArea (r):
      pi=3.14 # constant
      A=pi * r * r
return A
```

Why do we need user defined functions?



Describe above math figures.
Write programs to calculate their areas?
Are formulas like functions?

One function output can be an input to another.

Can you write a recursive program for oddtorial ?
Use what you learned from factorial program.

Variable Scope - Can you see me or not?
Just because we **declare** a variable doesn't mean that it can be accessed from anywhere in our code. Each variable has its own **area of visibility** in your code.

A **global variable** is declared **outside** the functions.
All **functions can access** and modify its value.

What is the following code output?

```
x= 'global variable'
def func():
    print("inside function")
    print(x)
func()
print("after function")
print(x)
```

x is not declared inside func(), therefore func() looks one-level up, finds x and print. x is visible globally.

A program has a function called **Subtractor**.
This function takes a number as input, subtract 1 from input value and prints new value and calls **Subtractor** again with new value as input.

Suggest a method to end the endless recursion.
Draw a flowchart for the program.
Write a python program and test with following inputs.
12, 8, 5, 4, 0 , -2, -12

Factorial !

Factorial of a whole number n,denoted by n!
n! is the product of all positive integers less than or equal to n. Factorial of 5 is 5 x 4 x 3 x 2 x 1.

!2	2 * 1		2
!3	3 * 2 * 1		
!4			

Explain following factorial function.

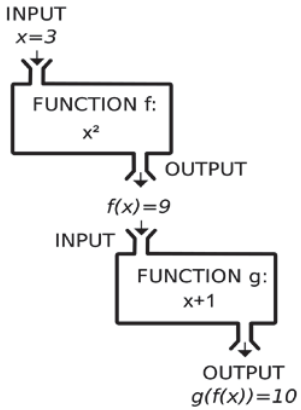
```
def factorial(n):
    if (n == 1):
        return 1
    else:
        return n * factorial(n-1)
```

How do you use it in a program?

Professor Crow's Oddtorial problem

If Professor Crow **define** oddtorial as the product of all positive odd integers less than the given number, find the oddtorial of 5, 6 and 7. *Tip : draw a flowchart.*

What happens here?



Function f receives **input x**

Function f **process x** and output x^2

Function g receives input x^2.

Function g process x^2 and output $x^2 + 1$

Draw a flowchart for above program.
Test your flowchart with following inputs.

x		2	3	4	0	-1	200
output							

Can you you **write a python program** for your flowchart?.

```
def : function2(Y)
       code
       return result2

def : function1(X)
       result1=function2(Y)
       return result1

Z=function1(X)
```

Is following can be a **valid program** for chef robot ?

```
recipe=”Chocolate Cake”
food=serve(cook(clean(buy(recipe))))
```

Which function get executed first?
What inputs each function take?
What is the process of each function?
What outputs may be returned by function?

The output from function can be a number, string or boolean value. It also can return a data set like a list. Function can also be designed not to return any value.

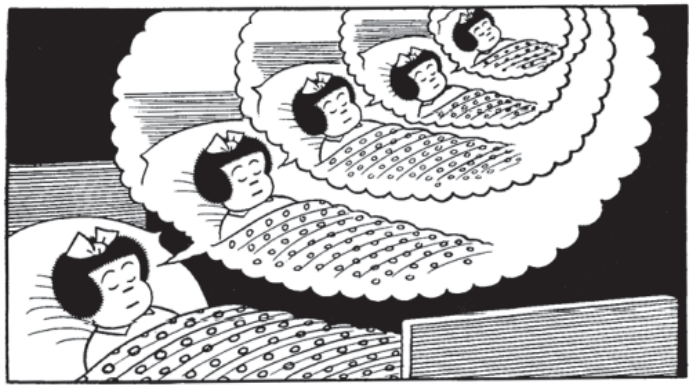
Can a function call it self?
Write following code in python and run.

```
def greetings(name):
        gtext="Welcome ." + name
        print (gtext)
        greetings(name)

name=input("Enter Name")
greetings(name)
```

What is happening?
Press Control + C to stop the program
Your greetings() function calls itself again and again, and get stuck in an endless process.
Recursion is like Nancy seeing a dream inside another

dream and so on.



How to return from recursive endless loops ?
You use a counter and pass it to each function call as an input, then increment it inside the function, and exit the function after number of recursive function calls.

```
def greetings(name,counter):
    if(counter > 10):
        return counter
    else:
        counter=counter+1
        gtext="Welcome ." + name
        print (gtext)
        greetings(name, counter)

name=input("Enter Name")
counter=0
greetings(name, counter)
```

Draw a flowchart and explain this process.

Recursion finds the solution to a problem is based on solving smaller instances of the same problem.