# functions

- Function is a grp of stmts to perform a specific task
- function breks a codes into small modules to look more organised
- code reuseability
- types of functions
  - builtin functions
  - user defined functions

```
In [1]: ## list of builtins or builtin functions
        dir(__builtins__)
```

```
Out[1]: ['ArithmeticError',
         'AssertionError',
         'AttributeError',
         'BaseException',
         'BlockingIOError',
         'BrokenPipeError',
         'BufferError',
         'BytesWarning',
         'ChildProcessError',
         'ConnectionAbortedError',
         'ConnectionError',
         'ConnectionRefusedError',
         'ConnectionResetError',
         'DeprecationWarning',
         'EOFError',
         'Ellipsis',
         'EnvironmentError',
         'Exception',
         'False',
         'FileExistsError',
         'FileNotFoundError',
         'FloatingPointError',
         'FutureWarning',
         'GeneratorExit',
         'IOError',
         'ImportError',
         'ImportWarning',
         'IndentationError',
         'IndexError',
         'InterruptedError',
         'IsADirectoryError',
         'KeyError',
         'KeyboardInterrupt',
         'LookupError',
         'MemoryError',
         'ModuleNotFoundError',
         'NameError',
         'None',
         'NotADirectoryError',
         'NotImplemented',
         'NotImplementedError',
         'OSError',
         'OverflowError',
         'PendingDeprecationWarning',
         'PermissionError',
         'ProcessLookupError',
         'RecursionError',
         'ReferenceError',
         'ResourceWarning',
         'RuntimeError',
         'RuntimeWarning',
         'StopAsyncIteration',
         'StopIteration',
         'SyntaxError',
         'SyntaxWarning',
         'SystemError',
         'SystemExit',
         'TabError',
         'TimeoutError',
         'True',
         'TypeError',
         'UnboundLocalError',
         'UnicodeDecodeError',
         'UnicodeEncodeError',
         'UnicodeError',
         'UnicodeTranslateError',
         'UnicodeWarning',
         'UserWarning',
         'ValueError',
         'Warning',
         'WindowsError',
         'ZeroDivisionError',
         '__IPYTHON__',
         '__build_class__',
         '__debug__',
         '__doc__',
         '__import__',
         '__loader__',
         '__name__',
         '__package__',
         '__spec__',
         'abs',
         'all',
         'any',
         'ascii',
         'bin',
         'bool',
         'breakpoint',
         'bytearray',
         'bytes',
         'callable',
         'chr',
         'classmethod',
         'compile',
         'complex',
         'copyright',
         'credits',
         'delattr',
         'dict',
         'dir',
         'display',
         'divmod',
         'enumerate',
         'eval',
         'exec',
         'filter',
         'float',
         'format',
         'frozenset',
         'get_ipython',
         'getattr',
         'globals',
         'hasattr',
         'hash',
         'help',
         'hex',
         'id',
         'input',
         'int',
         'isinstance',
         'issubclass',
         'iter',
         'len',
         'license',
         'list',
         'locals',
         'map',
         'max',
         'memoryview',
         'min',
         'next',
         'object',
         'oct',
         'open',
         'ord',
         'pow',
         'print',
         'property',
         'range',
         'repr',
         'reversed',
         'round',
         'set',
         'setattr',
         'slice',
         'sorted',
         'staticmethod',
         'str',
         'sum',
         'super',
         'tuple',
         'type',
         'vars',
         'zip']
```

# user defined functions

## user syntax in c

```
function fname(){
    condition or statements to execute
}


### syntax in python
def fname():
    conditions or stmts
    return
fname()
```

- advantages of a function:-
  - making large code into small modules
  - reuse of code in a function by calling its function name
- types of arguments in functions:-
  - required arguments
  - keyword arguments
  - default arguments
  - variable length arguments

```
In [6]: # sum of 2 numbers
        a=54
        b=10
        sum([a,b])

Out[6]: 64
```

```
In [8]: # find the max number
        a=[1,2,3,4,5,34764]
        max(a)

Out[8]: 34764
```

In [9]:
```
#find the minimum numbers
a=[1,2,3,4,5,6]
min(a)
```
Out[9]:
```
1
```

In [12]:
```
# find the length of a function
#a=[1,2,3,4,5]
a=('gowribindu')
len(a)
```
Out[12]:
```
10
```

In [20]:
```
#required argument
def add(a,b):
    c=a+b
    return c
a=int(input('enter a value'))
b=int(input('enter b value'))
add(a,b)
```
```
enter a value3
enter b value4
```
Out[20]:
```
7
```

In [23]:
```
# keyword arguments
def key(str):
    print(str)
key(str=123)
```
```
123
```

In [25]:
```
#keyword argument
def keyw(name,clg):
    print('name:',name)
    print('clg:',clg)
keyw(name='hima',clg='aits')
```
```
name: hima
clg: aits
```

In [30]:
```
# default argument
def default (a=10,b='abc'):
    print(a,b)
default(a,b)
```
```
3 4
```

In [31]:
```
#default arguments
def default(1,r=1):
    print
```

```
  File "<ipython-input-31-a1cebb90eff0>", line 2
    def default(1,r=1):
               ^
SyntaxError: invalid syntax
```

## task

- print n natural numbers using functions
- check weather given num is prime r not

In [ ]:
```
# n odd numbers using functions
n=int(input('enter the value'))
def odd(n):
    for i in range(1,n+1):
        if i%2 != 0:
            print(i,end=' ')
    return
odd(n)
```

In [3]:
```
n=int(input('enter the values'))
def prime(n):
    c=0
    for i in range(1,n+1):
        if n%i==0:
            c=c+1
    if c==2:
        print(n,'is prime')
    else:
        print(n,'is not prime')
prime(n)
```

```
enter the values9
9 is not prime
```

In [ ]: