Queen Mary
University of London

QHP4701
Introduction to Data Science Programming

Function: Program Development

Lecturer: Nikesh Bajaj, PhD
School of Physical and Chemical Sciences
http://nikeshbajaj.in

Lecture Outline

Control flow tools

- Condition with sequences
- Condition with sequences with ‘in’ operator

Function Definition

- Function Definition and arguments
- Function Definition: calling
- Function Definition: arguments/returns
- Modular Approach in Programming
- Function Definition: Recursion

Conditions with sequences: List, Tuple

- Sequence data types in python are: ***list, tuple, dict***, and ***set***.
- List
  - For given two lists, we can test if they are equal or not *using '=='*

```
x = [3, 10, 10, 50]
y = [3, 10, 50]

if x==y:
   print('x and y are same')
```

  - Operators like >, <, >=, <= are not applicable for lists ***or strings***
- Tuple: Two tuples can be testes for equality (same as list)

Conditions with sequences: Dictionaries

- Dictionaries: dict
  - For given two dictionaries, we can test if they are equal or not *using* '=='

```
x = {'A':1, 'B':2, 'C':3}
y = {'A':1, 'B':2, ‘C’:4}

if x==y:
   print(‘x and y are same’)
```

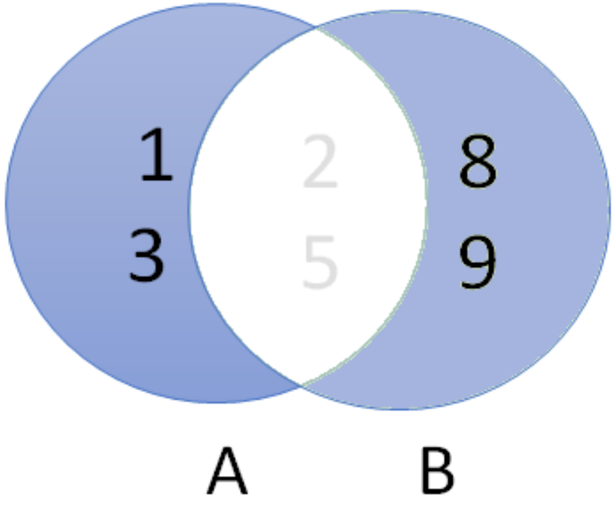
  - We can also check two dictionaries have same length and same keys

```
if len(x) == len(y):
   print(‘x and y have same length’)
```

```
if x.keys() == y.keys():
   print(‘x and y have same keys’)
```

Conditions with sequences: Sets

- Sets: set
  - For given two sets, we can test equality, and exclusion '==', '>', '<',
    - A == B   if A is equal to B
    - A < B   if A is proper subset of B
    - A <= B   if A is subset of B
    - A > B.   If A is proper superset of B
    - A >= B.   If A is superset of B




```
x = set([1,2,2,2,3])
y = set([2,1,1,3])
if x == y:
   print('x and y are same')
```

Conditions with sequences: Sets

- Set: question

```
A = set([1,2,2,2,3])

B = set([2,1,1,3])

C = set([2,1,3,4,4])
```

- A == B   if A is equal to B
- A < B   if A is proper subset of B
- A <= B   if A is subset of B
- A > B.   If A is proper superset of B
- A >= B.   If A is superset of B

- Check which one is subset of which and proper subset of which
  - Is A subset of B and C?
  - Is B subset of A and C?
  - Is C subset of A and B?

Lecture Outline

Control flow tools

- Condition with sequences
- Condition with sequences with ‘in’ operator

Function Definition

- Function Definition and arguments
- Function Definition: calling
- Function Definition: arguments/returns
- Modular Approach in Programming
- Function Definition: Recursion

Conditions with sequences: 'in'

$x \in Y$

- In Python, **in** operator allows you to test, whether an element is in given sequences, which can be a list, tuple, set or keys of a dictionary

```
x1 = [1,2,3]
x2 = (1,2,3)
x3 = set([1,2,2,2,3])
x4 = {1:'A', 2:'B', 3:'C'}
```

```
if 1 in x1:
    print('1 is in x1')

if 1 in x2:
    print('1 is in x2')

if 1 in x4.keys():
    print('1 is in x4')
```

```
if 1 in x4:
    print('1 is in x4')
```

Conditions with sequences: `'in'`

$x \in Y$

- **in** operator is useful to avoid errors and create dynamic sequences
  - Before looking index of an element, test if it is in the list
  - Create new key-value pair in dictionary, only if it doesn't exist

```
x = ['A', 'B', 'C', 'G']

if 'K' in x:
    idx = x.index('K')
    print('K is in x at location',idx)
```

```
y = {'A':1, 'B':2, 'C':3}

if 'D' not in y:
    y['D'] = 4
```

Conditions with sequences: `'in'`

$x \in Y$

- Question
- Given a sentence

  S = ‘set theory is one of the greatest achievements of modern mathematics’

- Create a dictionary that tells the frequency of all the vowels in the sentence S

The output should look something like this:

```
freq_vowels = {'a':10, 'e':12, 'i':2, 'o':5, 'u':0}
```

Lecture Outline

Control flow tools

- Condition with sequences
- Condition with sequences with ‘in’ operator

Function Definition

- Function Definition and arguments
- Function Definition: calling
- Function Definition: arguments/returns
- Modular Approach in Programming
- Function Definition: Recursion

Defining a function

- To reuse a block of code, a function can be created, then it can be called anywhere in the script.

```
def myfun():
    print('Welcome to my code')


myfun()
```

```
def fun_name():
 …
 do something with x
 …
```

Defining a function
Arguments and returns

- A function can have input and output arguments.


```python
def myfun(x, y):
    z = (x**2 - y**2)
    return z


Z = myfun(2,3)
Z = myfun(x=2, y=3)
```

```python
def myfun(x,y):
    …
    do something with x
    …
    return z
```

Lecture Outline

Control flow tools

- Condition with sequences
- Condition with sequences with ‘in’ operator

Function Definition

- Function Definition and arguments
- Function Definition: calling
- Function Definition: arguments/returns
- Modular Approach in Programming
- Function Definition: Recursion

Defining a function
Calling a function

```
def myfun(x, y):
    z = (x**2 - y**2)
    return z
```

- Calling a function

- Positional arguments: the order the position should be same as function definition

```
Z = myfun(2,3)     #here x=2, y=3
Z = myfun(3,2)     #here x=3, y=2
```

- Argument with names: while passing arguments with names, the order doesn't matter

```
Z = myfun(x=2, y=3)
Z = myfun(y=3, x=2)
```

Defining a function: Question

- Question
- Create a function that returns grade of student for given marks M according to table

Marks	Grade
Above 90	A
Between 80 to 90	B
Between 70 to 80	C
Between 50 to 70	D
Below 50	F

```
def Grading(M):
    pass

    return G
```

Grad = Grading(M)

Defining a function: Question

- Question

- Create a function that add all the numbers in a list, and avoid None and any strings

For example

X = [1, 2, 3, None, 0, 4 ,2 , None, ‘A’]

Z = mySum(X)

Z = 12

```
def mySum(X):
    pass


    return Z
```

Defining a function : Question

- Question

- Create a function that returns a dictionary that has the frequency of each character (a-z) for given a string excluding space, number and any symbols

For example

- S = ‘set theory is one of the greatest achievements of modern mathematics’

Freq = Freq_Char(S)

```
def Freq_Char(S):
    Freq = {}
    pass


    return Freq
```

Lecture Outline

Control flow tools

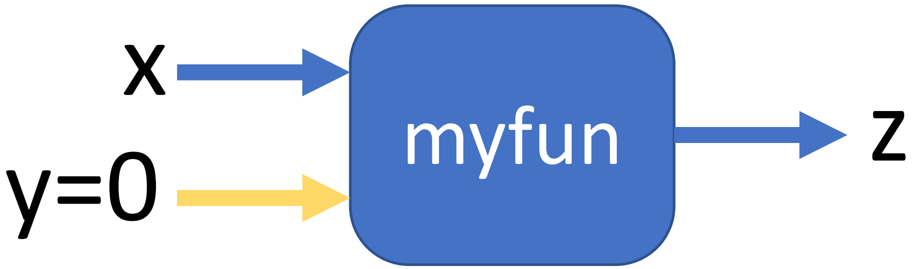
- Condition with sequences
- Condition with sequences with ‘in’ operator

Function Definition

- Function Definition and arguments
- Function Definition: calling
- Function Definition: arguments/returns
- Modular Approach in Programming
- Function Definition: Recursion

Defining a function
Default arguments

- A function can inputs with default values.

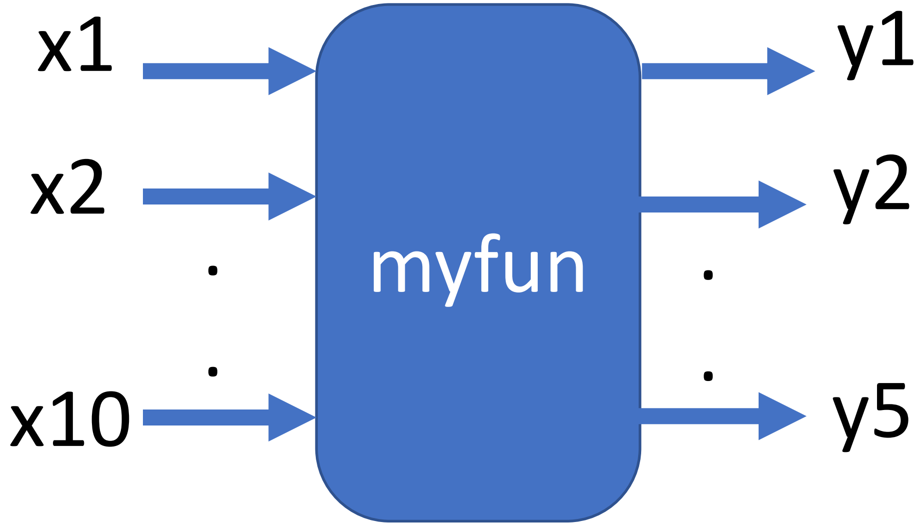



```
def myfun(x, y=0):
    z = (x**2 - y**2)
    return z


Z = myfun(2)
Z = myfun(2, y=3)
z = myfun(x=2,y=3)
z = myfun(y=3,2)
```

Defining a function
Multiple input/output

- A function can multiple inputs and outputs.




```
def myfun(x1, x2, x3=0):
    y1 = (x1**2 + x2**2 + x3**2)
    y2 = (x1**2 – x2**2 - x3**2)
    return y1, y2

y1,y2 = myfun(2,3,4)
```

Defining a function : Question

- Question

```
def sum_cat(X):
    y1,y2,y3 = 0,0,0
    pass

    return y1,y2,y3
```

Given a list X as follow.

X = [0.1,3.5,5.0,10,0.5,0.3,None,0.1,6.0,10.4,6.2,7,8.9, 1]

Write a piece of code to

- Add all the numbers in X2 which are **below 1 (including 1)**, save them to variable name 'y1’
- Add all the numbers in X2 which are **above 1 and below 6 (including 6**), save them to variable name 'y2’
- Add all the numbers in X2 which are **above 6 (excluding 6)**, save them to variable name 'y3'

Defining a function
Multiple *return* statement

- A function can multiple return statements, however function exist when it seems return.

```
def myfun(x, y=0):
    if x==0:
        return 0
    z = (x**2 - y**2)
    return z

z= myfun(0)
```

*It is a **good practice** to write docstring for a defined function.*

Defining a function: Question

- Question
- Use multiple return statement in following


```
def Grading(M):
    if M>=90:
        return ‘A’

    return ‘F’
```

Marks	Grade
Above 90	A
Between 80 to 90	B
Between 70 to 80	C
Between 50 to 70	D
Below 50	F

Grad = Grading(M)

Defining a function
Docstring

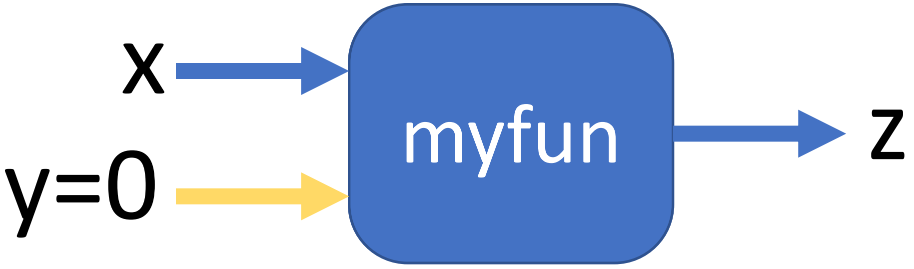
- A function can docstring that explains the functions operation.

which can be accessed by a user using `help(myfun)`

*It is a **good practice** to write docstring for a defined function.*

```
def myfun(x, y=0):
    '''
    This function computes x^2 – y^2
    '''
    z = (x**2 - y**2)
    return z

help(myfun)
```



Lecture Outline

Control flow tools

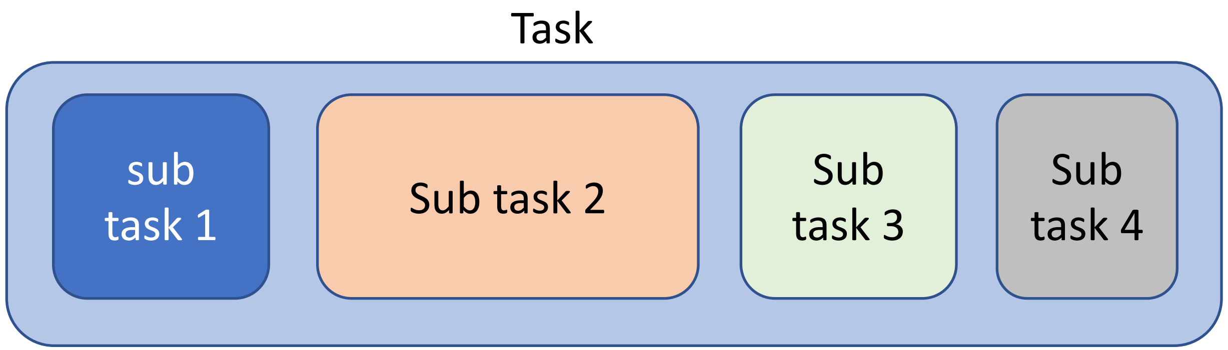
- Condition with sequences
- Condition with sequences with ‘in’ operator

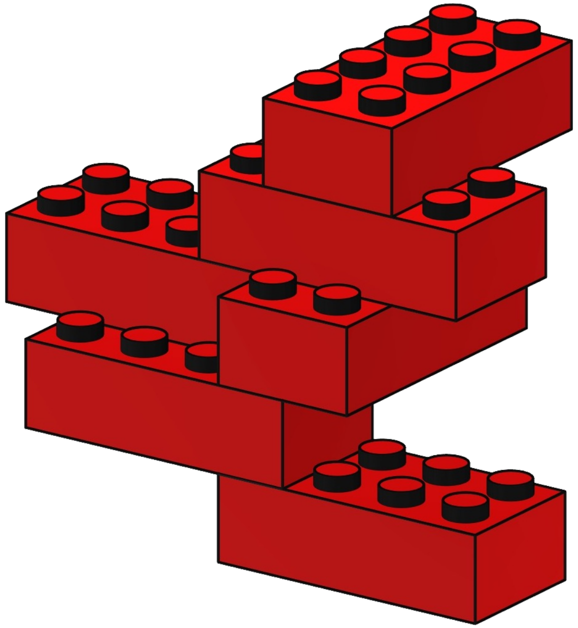
Function Definition

- Function Definition and arguments
- Function Definition: calling
- Function Definition: arguments/returns
- Modular Approach in Programming
- Function Definition: Recursion

Modular Approach in Programming

- Modular approach to any task is a process of sub-dividing a bigger task into smaller one.
- Solving/completing smaller tasks first to complete the big task.
- It is a ***good practice*** to break a task in multiple smaller tasks.
- Each sub-task handles a specific operation





Defining a function
Calling function in another function

- A function can be called in another function.

```
def is_even(x):
     if x%2==0:
         return True
    return False

def even_sum(X):
    c = 0
    for x in X:
         if is_even(x)
              c = c +x
    return c
```

Modular Approach in Programming

*It is a **good practice** to break a task in multiple smaller tasks.*

Defining a function: Question

- Question
- Write a function to find ***N*** prime numbers starting from 2
- Before that write a function to test if given number if prime

```
def prime_numbers(N):
    p = []
    pass #complete the code
     if is_prime (x):
        p.append(x)
    return p
```

```
def is_prime (x):
    pass #complete the code
    if cond:
        return True
    return False
```

P = prime_numbers(10)

P=[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]

Lecture Outline

Control flow tools

- Condition with sequences
- Condition with sequences with ‘in’ operator

Function Definition

- Function Definition and arguments
- Function Definition: calling
- Function Definition: arguments/returns
- Modular Approach in Programming
- Function Definition: Recursion

Defining a function
Calling function in itself

Recursion

- A function can be called in by itself.

For example

- How do you compute factorial of n?

n! = n*(n-1)*(n-2) ….. 3*2*1

(n-1)!

n! = n*((n-1)!)

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

Defining a function: Question

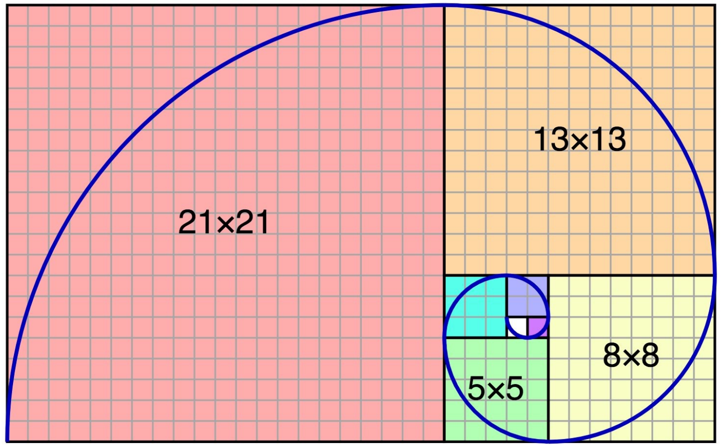
- Question
- Write a function to compute **Fibonacci** series

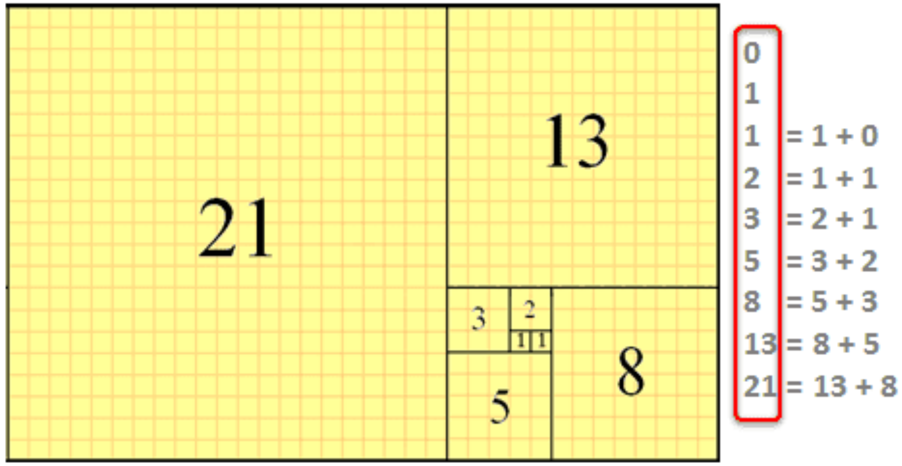
$$F_n = F_{n-1} + F_{n-2}$$

$F_0 = 0$

$F_1 = 1$

[0, 1, 1 , 2, 3, 5, 8, 13, 21 …]





- Next !!!
    - 3.4: Visualisation with Matplotlib

Queen Mary
University of London