PYTHON

From Simple to Complex
With
Examples

AYESHA NOREEN

Bachelor's in Software Engineering,

Master's in Computer Science

from COMSATS University, Islamabad

NOTE!!!

In these notes Screenshots of practice examples and coding are added. The code files are also available in code folder that contain .ipynb files that are created on Jupyter notebook.

Chapter7
Functions in Python

- **Functions**

Syntax:

```
Def  function_name(parameters):     #function defination
       #here working of function
       Return                                #function return
function_name(arguments)            #function call
```

- **Parameter VS Argument**

A parameter is the variable listed inside the parentheses in the function definition.

An argument is the value that is sent to the function when it is called.

- **TODO Task**

Define a function that take two numbers from user and return its sum and also concat two strings with same function.

```
def total(a,b):
    return a+b
    print(a+b)  #we can also print directly without return
num1=int(input("Enter number1:"))
num2=int(input("Enter number2:"))
Sum=total(num1,num2) #function call and store result in a variable
print(total(num1,num2)) #we can also print directky
print(f"Sum of numbers is:{Sum}")
print(total("Ayesha ","noreen")) # we can also pass strings total function concate two strings
```

```
Enter number1:5
Enter number2:5
10
Sum of numbers is:10
Ayesha noreen
```

- **TODO Task**

Define a function that return last character of a string.

```
def last_char(name):
    return name[-1]
print(last_char("Ayesha")) #we can pass string in parameters
n=input("Enter any name whose last character you want to print: ") #we can also take input
a=last_char(n) #than pass input in parameter
print(f"last charcater of {n} is {a}")
#last_char(9)#if we pass any number it gives error bcz function return a string index not number
```

```
a
Enter any name whose last character you want to print: Allah
last charcater of Allah is h
```

- **TODO Task**

Define a function that takes two number from user and check which number is greater.

```
#take two number from user as input than show which number is greater
def greater(num1,num2):
    if num1>num2:
        return "number1 is greater than number2"
    return "number2 is greater than number1"
n1=int(input("Enter number1:"))
n2=int(input("Enter number2:"))
print(greater(n1,n2))
```

```
Enter number1:7
Enter number2:34
number2 is greater than number1
```

- **TODO Task**

Define a function that checks either a number is even or odd.

```
def even_odd(num):
    if num%2==0:
        return "Even"
    else:
        return "odd" #function can also return a string
number=int(input('Enter number: '))
print(even_odd(number))
```

```
Enter number: 6
Even
```

- # Function inside function

```
def greater(num1,num2):
    if num1>num2:
        return num1
    return num2
def greatest(n1,n2,n3):
    big=greater(n1,n2)
    return greater(big,n3)
a=int(input("Enter number1:"))
b=int(input("Enter number2:"))
c=int(input("Enter number3:"))
print(f"Greatest number is:{greatest(a,b,c)}")
```

```
Enter number1:12
Enter number2:56
Enter number3:23
Greatest number is:56
```

- **TODO Task**

Take any string and print either it is palindrom or not

PALINDROM mean forwarding and reversing strings are same. e.g mom, madam etc.

```
# Take any string and print either it is palindrom or not
def is_palindrom(name):
      if name[::-1]==name:
         return "Name is palindrom"
      else:
         return "Name is not palindrom"
n=input("Enter any name:")
print(is_palindrom(n))
```

```
Enter any name:ayesha
Name is not palindrom
```

- **TODO Task**

Take a number from user and print fabbnocci series.

```
#fabinacci series  0 1 1 2 3 5 8 13 21 34.............
def fabnnocci(n):
    a=0
    b=1
    if n==1:
        print(a)
    elif n==2:
        print(a,b)
    else:
        print(a ,b, end=" ")
        for i in range(n-2):
            c=a+b
            a=b
            b=c
            print(b, end=" ")
num=int(input("Enter how many numbers of fabnocci series you want to print:"))
fabnnocci(num)
```

```
Enter how many numbers of fabnocci series you want to print:5
0 1 1 2 3
```

- **Function as an argument**

```
#function as an argument
def square(num):
    return num**2
L=[1,2,3,4,5]
print(list(map(square,L)))
#this is build in map function that print square as[1,4,9,16,25]

def my_map(func,l):
    new=[]
    for i in l:
        new.append(func(i))
    return new
print(my_map(square,L)) #it also prints [1,4,9,16,25]
#we can also do this by list comprehension
def my_map2(fun,l):
    return [fun(item) for item in l]
print(my_map2(lambda x:x**3,L)) #it prints [1, 8, 27, 64, 125]
```

```
[1, 4, 9, 16, 25]
[1, 4, 9, 16, 25]
[1, 8, 27, 64, 125]
```

- **Function returning a function**

Function returning a function also known as first class function and closure.

```
#function returning a function
def outer(msg):
    def inner():
        print(f"Message is:{msg}")
    return inner
print(outer("hello")) #give outer object bcz outer return a function
a=outer("hello")
print(a()) #it returns message is hello
```

```
<function outer.<locals>.inner at 0x00000180A1B1CD30>
Message is:hello
None
```

- **TODO Task**

Calculate either square or cube or raised to power of any number by using a single function.

```
def to_power(x):
    def cal_power(n):
        return n**x
    return cal_power
cube=to_power(3)
print(cube(5)) #it prints cube of 5=125
square=to_power(2)
print(square(4)) #it print 4 square=16
power_five=to_power(5)
print(power_five(2)) #it prints 2 raised to power 5=32
```

```
125
16
32
```

• Function parameters

Function have following parameters

- Parameters (normal parameters)
- Default parameters (set default values of parameters)
- Arguments (*args) return a tuple
- Keyword arguments (**kwargs) return a dictionary

If we want to use all than there order is PADK:

Function_name(parameters,*args, default parameters,**kwargs):

Note!!!

In this chapter we cover only normal and default parameters. I will explain *Args and **kwargs later (in chapter16) after having understanding about list, tuples, sets and dictionaries etc.

○ Normal parameters

Normal parameters are user defined parameters whose value can be taken as input from user or pass values during function call.

```
#take three numbers than print which is greatest
def greatest(n1,n2,n3):
    if n1>n2 and n1>n3:
        return "number1 is greatest"
    elif n2>n1 and n2>n3:
        return "number2 is greatest"
    else:
        return "number3 is greatest"
print(greatest(10000,200,30)) #pass values during call
n1=int(input('Enter number1: '))
n2=int(input('Enter number2: '))
n3=int(input('Enter number3: '))
print(greatest(n1,n2,n3))  #take values from user
```

```
number1 is greatest
Enter number1: 50
Enter number2: 100
Enter number3: 150
number3 is greatest
```

○ Default parameters

If we want to add any default value of some parameters than place these parameters at last because otherwise it gives error.(e.g. if 1st parameter is default and 2nd is normal than error).

```
def fun(first_name="unknown",last_name="unknown",age="none"):
    print(f"your first name is:{first_name}")
    print(f"your last name is:{last_name}")
    print(f"your age is:{age}")
fun("Ayesha","Noreen",22)
fun()
```

```
your first name is:Ayesha
your last name is:Noreen
your age is:22
your first name is:unknown
your last name is:unknown
your age is:none
```

• Global and local variables

Global variables are defined outside of a function and local variables are defined inside a function. Scope of global variable is within the whole program while scope of local variable is only within function in which it is defined.

```
x=10  #global variable
def fun():
    global x
    x=5    #local variable
    return x
print(x)          #print global x which is 10 bcz fun is not call yet
print(fun())      #print 5 bcz we use global x in fun which changes the value of x=5
print(x)          #print 5
```

```
10
5
5
```

- **Pre-Defined Functions**

Pre-defined functions are those functions which are defined by developers for our easiness and we can use them. For Example,

len() return length of string and count() function return count of character in a string etc. There are lot of pre-defined functions in python which we can use by importing there module or package.

i.e. math module contain all math functions

numpy module contain numerical function

scipy contain scientific functions

pandas contain dataframe function etc.

- **Some mostly used Pre-Defined Functions**
- **enumerate() function**

Enumerate function contain position of variables it returns position.

```
names=['Ayesha','Noreen','Sana','Adnan']
pos=0
for name in names:
    print(f"{name} is at position or at index number {pos}")
    pos+=1
#finding position with the use of enumerate function
for i,j in enumerate(names):
    print(f"{j} is at position or at index number {i}")
```

```
Ayesha is at position or at index number 0
Noreen is at position or at index number 1
Sana is at position or at index number 2
Adnan is at position or at index number 3
Ayesha is at position or at index number 0
Noreen is at position or at index number 1
Sana is at position or at index number 2
Adnan is at position or at index number 3
```

- **TODO Task**

Define a function that take two parameters one is list of strings and other is string than find string in list of strings if find out than return position otherwise return -1by using enumerate function.

```
def position_finder(l,string):
    for i,j in enumerate(l):
        if j==string:
            return i
    return -1
names=['ayesha','noreen','sana','adnan']
print(position_finder(names,'sana'))
```

```
2
```

• map() function

Map() function is used to map values in a function. Function is either user defined or pre defined and values are of any data type.

```
def square_calculator(num):
    return num**2
#map a list numbers into a square_calculator function
#convert into list and save in squares list
numbers=[1,2,3,4,5]
squares=list(map(s,numbers))
print(squares)
#we can also use a pre-defined function in map function
List=['abc','cba','ad']
length=list(map(len,List))
#we can apply loop on map object
print(length)
for index in length:
    print(index)
```

```
[1, 4, 9, 16, 25]
[3, 3, 2]
3
3
2
```

• filter() function

filter() function is used to filter the required values.

```
#filter function which filters even numbers.
def is_even(num):
    return num%2==0
numbers=[1,2,4,5,8,3]
evens=tuple((is_even,numbers))
print(evens)
#we can also perform above function by use of lambda expression as
evens=tuple(filter(lambda i:i%2==0,numbers))
print(evens)
for i in evens:
    print(i)
#if we remove tuple from above method than loop is run only one time
#we can also do above function by using list comprehension as
even=[i for i in numbers if i%2==0]
print(even)
```

```
(<function is_even at 0x000001B6D1AF15A0>, [1, 2, 4, 5, 8, 3])
(2, 4, 8)
2
4
8
[2, 4, 8]
```

Iterables and iterators

- **Iter() and next() function**

Let, Number=[1,2,3,4,5]

Here, Number is iterables because we can apply loop on it. As,

For i in Number:

    print(i)

Each time i prints its value iter() function is called and after it next() function is called which increment in loop.

Iterables and iterators

```python
numbers=[1,2,3]
returned_next=iter(numbers) #if we donot call this iter function and directly call next than give error
print(returned_next) #<list_iterator object at 0x000000142E6D2B00>
print(next(returned_next)) #1
print(next(returned_next)) #2
print(next(returned_next)) #3
#print(next(returned_next)) #error
squares=map(lambda i:i**2,numbers)
print(squares)
print(next(squares))
print(next(squares))
print(next(squares))
#in above method we directly call next fuction iter is not call than it print squares not error
#so this is iterator
```

```
<list_iterator object at 0x000001B6D1238B50>
1
2
3
<map object at 0x000001B6CFBC0430>
1
4
9
```

• zip() function

Zip() function is used to zipify two or more iterables (list,tuple,set,dict) and return a tuple than we can convert it into dictionary if each tuple have two items.

```
reg_no=(1,2,3,4)
names={'ayesha':4,'zneesha':2,'sehrish':3,'zoya':4}
cgpa=['3.2','3.6']
city=['lahore','okara','sahiwal','patoki']
print(list(zip(reg_no,names.items())))
print(list(zip(reg_no,names.values())))
print(list(zip(reg_no,names))) #by default take keys
#if length is not same than it discard extra elements
print(list(zip(names,cgpa)))
#we can also convert above list into dictionary as
print(dict(zip(names,cgpa)))
#we can also use more than 2 lists
print(list(zip(reg_no,names,cgpa,city)))
```

```
[(1, ('ayesha', 4)), (2, ('zneesha', 2)), (3, ('sehrish', 3)), (4, ('zoya', 4))]
[(1, 4), (2, 2), (3, 3), (4, 4)]
[(1, 'ayesha'), (2, 'zneesha'), (3, 'sehrish'), (4, 'zoya')]
[('ayesha', '3.2'), ('zneesha', '3.6')]
{'ayesha': '3.2', 'zneesha': '3.6'}
[(1, 'ayesha', '3.2', 'lahore'), (2, 'zneesha', '3.6', 'okara')]
```

• * operator in zip() function

*operator in zip function is used for separation or unpacking of different list from a single list.

```
#* operator in zip() method
numbers=[(1,2,0,1.1),(3,4,0,1.2),(5,6,0,1.3)]
odd,even,zeros,floats=list(zip(*numbers))
print('odds are:',list(odd))
print("evens are:",list(even))
print("zeros are:",list(zeros))
print("floats are:",list(floats))
```

```
odds are: [1, 3, 5]
evens are: [2, 4, 6]
zeros are: [0, 0, 0]
floats are: [1.1, 1.2, 1.3]
```

- **TODO Task**

Print max number from tuple that zip() return.

```
#print max element from tuple that zip() return
odd=[1, 3, 5]
even=[2, 4, 6]
zeros=[0, 0, 0]
floats=[1.1, 1.2, 1.3]
new_list=[]
for i in zip(odd,even,zeros,floats):
    new_list.append(max(i))
print(new_list)
```

```
[2, 4, 6]
```

- **TODO Task**

A function that return average of elements of each tuple that zip() return

```
l1=[1,2,3,4]
l2=[5,6,7,8]
l3=[9,3,5,7]
l=list(zip(l1,l2,l3))
s=[]
for i in l:
    s.append((sum(i))/3)
print(s)
#we can also define above function in 2 lines as
a= lambda *args:[sum(i)/len(i) for i in zip(*args)]
print(a([1,2,3,4],[5,6,7,8],[9,3,5,7]))
```

```
[5.0, 3.6666666666666665, 5.0, 6.333333333333333]
[5.0, 3.6666666666666665, 5.0, 6.333333333333333]
```

• All() and any() function

All() function returns true if all values are true and false if all values are false.
any()function returns true if any value is true and false if any value is false.

```
#any() and all() function
l1=[2,4,6,10,8]
print(all(i%2==0 for i in l1))#it prints true because all l1 items are even
l1=[2,4,3,10,8]
print(all(i%2==0 for i in l1))#it prints false because not all l1 items are even
l1=[1,5,3,7,8]
print(any(i%2==0 for i in l1))#it prints true because one even is present in l1
l1=[1,3,5,9,7]
print(any(i%2==0 for i in l1))#it prints false because not any number is even
```

```
True
False
True
False
```

- **TODO Task**

Define a function which check the type of input and than print its sum if input type is int or float

```
def total(*args):
    s=0
    if all([(type(a)==int or type(a)==float) for a in args]):
        for i in args:
            s+=i
        return s
    else:
        return "wrong input"
print(total(1,7,8,9,5,10))
```

```
40
```

- **Advanced min() and max() function**

```
#A function which return max string with respected to len of string
numbers=[1,5,6,10,7,8]
print(max(numbers)) #print 10
print(min(numbers)) #print 1
names=['ayesha','sana','adnan']
print(max(names)) #print sana because s is max than a
#but if we want max string w.r.t length of string,
print(max(names, key=lambda item:len(item))) #print ayesha
```

```
10
1
sana
ayesha
```

• Advanced min() and max() function

If we have a list of students and this list have many dictionaries if we want to print only one dictionary that have highest score

```
students=[
{'name':'ayesha','score':98,'age':22},
{'name':'areesha','score':80,'age':20},
{'name':'sana','score':90,'age':24}
]
print(max(students,key=lambda item:item.get('score')))
#above method print whole dictionary if we want to print only name
print(max(students,key=lambda item:item.get('score'))['name'])
#if we have a dictionary which contain many key value pairs each value
# have a dictionary and we want to calculate max and min from this
students2={
'ayesha':{'score':78,'age':22},
'sana':{'score':80,'age':14},
'adnan':{'score':98,'age':6}
}
print(max(students2,key=lambda item:students2[item]['score']))
```

```
{'name': 'ayesha', 'score': 98, 'age': 22}
ayesha
areesha
adnan
ayesha
```

• Advanced sorted() function

```
#if we have a list and we want to sort items of list than we apply sort method
names=['noreen','adnan','sana','ayesha']
names.sort()
print(names) #it prints ['adnan', 'ayesha', 'noreen', 'sana']
#if we want to sort items of set than sort method cannot be applicable on set
#we can apply sorted method
names2={'ayesha','sana','adnan','noreen'}
#names2.sort() error
names3=sorted(names2)
print(names2) #set is immutable so does not change{'ayesha', 'sana', 'adnan', 'noreen'}
print(names3) #['adnan', 'ayesha', 'noreen', 'sana']
names4=('farooq','arfan','umar','rehman')
#names4.sort() error bcz also on tuples cannot apply sort
names5=sorted(names4)
print(names4) #('farooq', 'arfan', 'umar', 'rehman')
print(names5) #['arfan', 'farooq', 'rehman', 'umar']
user_info=[{'name':'ayesha','age':22,'city':'kamalia'},
{'name':'noreen','age':20,'city':'sahiwal'},
{'name':'sana','age':28,'city':'karachi'}]
print(sorted(user_info,key=lambda item:item['age']))
print(sorted(user_info,key=lambda item:item['age'],reverse=True))
```

• Doc string

```
def add(a,b):
    '''This is Doc string!!!!
    This function add two numbers
    '''
    return a+b
print(add(5,8)) #this show 13
print(add.__doc__) #this prints doc string
print(sum.__doc__) #this show doc string of build in sum function
print(max.__doc__) #this show doc string of build in max function
```

```
13

This is Doc string!!!!
This function add two numbers

Return the sum of a 'start' value (default: 0) plus an iterable of numbers

When the iterable is empty, return the start value.
This function is intended specifically for use with numeric values and may
reject non-numeric types.
max(iterable, *[, default=obj, key=func]) -> value
max(arg1, arg2, *args, *[, key=func]) -> value

With a single iterable argument, return its biggest item. The
default keyword-only argument specifies an object to return if
the provided iterable is empty.
With two or more arguments, return the largest argument.
```

• Help() function

Help() function provides meaningful information about function. Functions are either user-defined or pre-defined.

```
#help() function
def add(a,b):
    return a+b
help(add)
help(sum)
# help(max)
# help(min)
# help(len)
```

```
Help on function add in module __main__:

add(a, b)
    #help() function

Help on built-in function sum in module builtins:

sum(iterable, /, start=0)
    Return the sum of a 'start' value (default: 0) plus an iterable of numbers

    When the iterable is empty, return the start value.
    This function is intended specifically for use with numeric values and may
    reject non-numeric types.
```

- # time() function

time() function provides time of execution of a function in seconds

```
#time() function
import time
t1=time.time()
def square(num):
    return num**2
n=int(input("Enter any number whose square you want to calculate:"))
print(f"Square is:{square(n)}")
t2=time.time()
t=t2-t1
print(f"Execution time of this program is:{t}sec")
```

```
Enter any number whose square you want to calculate:6
Square is:36
Execution time of this program is:4.182973384857178sec
```