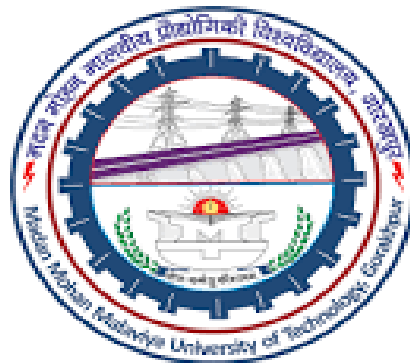


# PROGRAMMING IN PYTHON

MCA-161A

4 Credits (3-0-2)

MCA 5<sup>th</sup> Sem (2020-21)



R K Dwivedi  
Assistant Professor  
Department of ITCA  
MMUT Gorakhpur

# UNIT II: Control Flow and Other Programming Concepts

## 1. Iterative Statements:

For Loops, While Loops, Break, Continue

## 2. Array:

Looping Array elements, Array methods

## 3. Functions:

Local and Global Variables,  
Defining and calling the function,  
Functions with arguments,  
Recursion



# 1. Iterative Statements



## 1. Iterative Statements

### For Loop



Result Size: 668 x 476

```
fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
    print(x)
```

```
apple  
banana  
cherry
```



Result Size: 668 x 476






```
for x in "banana":  
    print(x)
```

```
b  
a  
n  
a  
n  
a
```



## 1. Iterative Statements






### For Loop (using 'break' and 'continue')



Result Size: 668 x 476

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
    if x == "banana":
        break
```

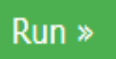




```
apple
banana
```



Result Size: 668 x 476

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        break
    print(x)
```

```
apple
```



Result Size: 668 x 476

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        continue
    print(x)
```

```
apple
cherry
```



## 1. Iterative Statements

### For Loop (using 'else')



Run »

Result Size: 668 x 476

```
li = [1,2,3,4,5]
for x in li:
    print(x)
    x += 1
else:
    print("x is no longer less than 6")
```

```
1
2
3
4
5
x is no longer less than 6
```



## 1. Iterative Statements

### For Loop (Nested Loop)



Run »

Result Size: 668 x 476

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]






for x in adj:
    for y in fruits:
        print(x, y)
```

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```



## 1. Iterative Statements






### While Loop



Result Size: 668 x 476

```
i = 1
while i < 6:
    print(i)
    i += 1
```

1  
2  
3  
4  
5



Result Size: 668 x 476

```
x=1
University="MMMUT"
while x<6:
    print(University)
    x=x+1
```






MMMUT  
MMMUT  
MMMUT  
MMMUT  
MMMUT





## 1. Iterative Statements






### While Loop (using 'break' and 'continue')



Result Size: 668 x 476

```
i = 1
while i < 6:
    print(i)
    if (i == 3):
        break
    i += 1
```

```
1
2
3
```



Result Size: 668 x 476

```
i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)

# Note that number 3 is missing in the result
```

```
1
2
4
5
6
```



## 1. Iterative Statements

### While Loop (using 'else')



Run »

Result Size: 668 x 476

```
i = 1
while i < 6:
    print(i)
    i += 1
else:
    print("i is no longer less than 6")
```

```
1
2
3
4
5
i is no longer less than 6
```



## 1. Iterative Statements

### While Loop (Nested Loop)



Run »

Result Size: 668 x 476

```
i = 1
while(i <= 3):
    j = 1
    while(j <= i):
        print('*', end = ' ')
        j = j + 1
    i = i + 1
    print()
```

```
*
* *
* * *
```



## 1. Iterative Statements

### Some Important Concepts of “Range” within “Loops”



Result Size: 668 x 476

```
#range()

for x in range (2,6):
    print(x)
print()

for x in range (2,30,3):
    print(x)
print()

for x in range (6):
    print(x)
print()
```

```
2
3
4
5

2
5
8
11
14
17
20
23
26
29

0
1
2
3
4
5
```



## 1. Iterative Statements

### Some Important Concepts of “Range” within “Loops”

...continued



Result Size: 668 x 476

```
#range()

for x in range (2,6):
    print(x, end=' ')
print()

for x in range (2,30,3):
    print(x, end=' ')
print()

for x in range (6):
    print(x, end=' ')
print()
```

```
2 3 4 5
2 5 8 11 14 17 20 23 26 29
0 1 2 3 4 5
```



## 2. Array



## 2. Array (Looping Array Elements and Array Methods)

...continued

Result Size: 668 x 476

```
fruits = ['apple', 'banana', 'cherry', 'banana']  
cars = ["Ford", "Volvo", "BMW"]
```

```
print(fruits)  
print(cars)
```

```
x = fruits.index("cherry")  
print(x)
```

```
x = fruits.count("banana")  
print(x)
```

```
x = len(cars)  
print(x)
```

```
for i in cars:  
    print(i)
```

```
cars.sort()  
print(cars)
```

```
fruits.reverse()  
print(fruits)
```

```
['apple', 'banana', 'cherry', 'banana']  
['Ford', 'Volvo', 'BMW']  
2  
2  
3  
Ford  
Volvo  
BMW  
['BMW', 'Ford', 'Volvo']  
['banana', 'cherry', 'banana', 'apple']
```



## 2. Array (Looping Array Elements and Array Methods)



Result Size: 668 x 476

```
fruits = ['apple', 'banana', 'cherry']
cars = ["Ford", "Volvo", "BMW"]

cars[0] = "Toyota"
print(cars)

fruits.insert(1, "orange")
print(fruits)

cars.append("Honda")
print(cars)

cars.extend(fruits)
print(cars)

cars.remove("Volvo")
print(cars)

cars.pop(5)
print(cars)

z=cars.copy()
z.clear()
print(z)
```

```
['Toyota', 'Volvo', 'BMW']
['apple', 'orange', 'banana', 'cherry']
['Toyota', 'Volvo', 'BMW', 'Honda']
['Toyota', 'Volvo', 'BMW', 'Honda', 'apple', 'orange', 'banana', 'cherry']
['Toyota', 'BMW', 'Honda', 'apple', 'orange', 'banana', 'cherry']
['Toyota', 'BMW', 'Honda', 'apple', 'orange', 'cherry']
[]
```





## 2. Array

### Some Important Concepts of "Strings"



Result Size: 668 x 476

```
a = "Hello, World!"
print(a[1])

b = "Hello, World!"
print(b[2:5])

b = "Hello, World!"
print(b[-5:-2])

a = "Hello, World!"
print(len(a))

a = "Hello, World!"
print(a.lower())

a = "Hello, World!"
print(a.upper())

a = " Hello, World! " #removes whitespaces from the beginning/end
print(a.strip())

a = "Hello, World!"
print(a.replace("H", "J"))

a = "Hello, World!" #splits string into substrings if it finds a separator
print(a.split(","))
```

```
e
llo
orl
13
hello, world!
HELLO, WORLD!
Hello, World!
Jello, World!
['Hello', ' World!']
```



## 3. Functions



### 3. Function

#### Defining and Calling a function








Result Size: 668 x 476

```
def func():  
    print("MCA Class, MMMUT Gorakhpur")  
  
func()
```

MCA Class, MMMUT Gorakhpur



### 3. Function Passing Arguments

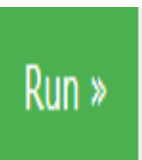






Result Size: 668 x 476

```
def func(fname, lname):  
    print(fname + ", " + lname)  
  
func("MCA Class", "MMMUT Gorakhpur")
```

MCA Class, MMMUT Gorakhpur

**If the number of arguments is unknown, add a \* before the parameter name.**



Result Size: 668 x 476

```
def func(*student):  
    print("The student of the year is " + student[2])  
  
func("Ram", "Shyam", "Mohan")
```






The student of the year is Mohan



### 3. Function Passing Arguments

...continued

You can also send arguments with the “key = value” syntax. Here, order of the arguments does not matter.

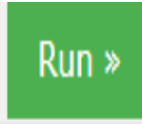






Result Size: 668 x 476

```
def func(student3, student2, student1):  
    print("The student of the year is " + student3)  
  
func(student1 = "Ram", student3 = "Mohan", student2 = "Shyam")
```

```
The student of the year is Mohan
```

If you do not know how many “keyword arguments” will be passed in the function, add \*\* before the parameter name in the function definition. Here, the function will receive a dictionary of arguments, and can access the items accordingly:



Result Size: 668 x 476

```
def func(**C):  
    print("This is " + C["C2"] + " class.")  
  
func(C1 = "B Tech", C2 = "MCA")
```

```
This is MCA class.
```






### 3. Function

#### Passing Arguments: Default Argument

...continued

If we call the function without argument, it uses the default value given during function definition.



Result Size: 668 x 476

```
def func(place = "Gorakhpur"):
    print("I am from " + place)

func("MMMUT")
func()
```

```
I am from MMMUT
I am from Gorakhpur
```








### 3. Function

#### Passing Arguments: Passing a List as an Argument

...continued

You can send any data types of argument to a function (string, number, list, dictionary etc.)



```
def func(food):  
    for x in food:  
        print(x)  
  
fruits = ["apple", "banana", "cherry"]  
  
func(fruits)
```

Result Size: 668 x 476

```
apple  
banana  
cherry
```



### 3. Function

#### “return” and “pass” statements



Result Size: 668 x 476

```
def func(x):  
    return 5 * x  
  
print(func(3))
```

15



Result Size: 668 x 476

```
def func():  
    pass
```

# An empty function definition like this, would raise an error without the pass statement





### 3. Function

#### Local and Global Variables



Result Size: 668 x 476

```
x = "awesome"                #Global Variable

def func():
    x = "fantastic"          #Local Variable
    print("Python is " + x)

func()                        #It will access x that is local to func()

print("Python is " + x)       #It will access global x
```

```
Python is fantastic
Python is awesome
```



### 3. Function Recursion



Result Size: 668 x 476

```
#Finding Facorial of a number using recursion
```

```
def fact(n):  
    if n == 1:  
        return n  
    else:  
        return n*fact(n-1)  
  
num = 3  
  
# check if the number is negative, zero or suitable positive  
if num < 0:  
    print("Sorry, factorial does not exist for negative numbers")  
elif num == 0:  
    print("The factorial of 0 is 1")  
else:  
    print("The factorial of", num, "is", fact(num))
```

```
The factorial of 3 is 6
```



# Some Programming Exercises



## Programming Exercise

1. Write a program to check a number **Even** or **Odd**.



Result Size: 668 x 476

#Checking a number Even or Odd

```
num = 5          #num = int(input("Enter No: "))
```

```
if (num % 2) == 0:  
    print(num, "is an EVEN No.")  
else:  
    print(num, "is an ODD No.")
```

5 is an ODD No.



## Programming Exercise

2. Write a program to **Weekdays** for the corresponding number.



Result Size: 668 x 476

```
#Print Weekdays for corresponding number
```

```
num=3                #num = int(input("Enter No: "))
if num==1:
    print('Sunday')
elif num==2:
    print('Monday')
elif num==3:
    print('Tuesday')
elif num==4:
    print('Wednesday')
elif num==5:
    print('Thursday')
elif num==6:
    print('Friday')
elif num==7:
    print('Saturday')
```

Tuesday



## Programming Exercise

3. Write a program for printing the **Star Triangle** using **Nested for**



Result Size: 668 x 476

#Printing Star Triangle using Nested For

```
for i in range(3):  
    for j in range(i+1):  
        print("*",end="")  
    print()
```

```
*  
**  
***
```



## Programming Exercise

4. Write a program to check a number **Prime** or not.



Result Size: 668 x 476

```
#Checking a number prime or not
```

```
num = 2                                #num = int(input("Enter No: "))
```

```
if num > 1:
    for i in range(2,num):
        if (num % i) == 0:
            print(num,"is not a prime number")
            break
    else:
        print(num,"is a prime number")
else:
    print(num,"is not a prime number")
```

```
2 is a prime number
```



## Programming Exercise

5. Write a program to check a number **Armstrong** or not.



Result Size: 668 x 476

```
#Checking a number Armstrong or not
```

```
num = 153          #num = int(input("Enter No: "))
```

```
sum = 0
```

```
temp = num
```

```
while temp > 0:
```

```
    digit = temp % 10
```

```
    sum += digit ** 3
```

```
    temp //= 10
```

```
if num == sum:
```

```
    print(num,"is an Armstrong number")
```

```
else:
```

```
    print(num,"is not an Armstrong number")
```





```
153 is an Armstrong number
```





## Programming Exercise

6. Write a program to **Reverse** a string



Run »

Result Size: 668 x 476

```
#Reverse a string  
  
s="MMUT"           #s = input("Enter the String: ")  
print(s[::-1])
```

```
TUMMM
```



## Programming Exercise

7. Write a program to check a string whether **Palindrome** or not.



Result Size: 668 x 476

```
string="MALAYALAM"
if(string==string[-1:-10:-1]):          #if(string==string[::-1]):
    print("The string is a palindrome")
else:
    print("The string is not a palindrome")
```

The string is a palindrome



## Programming Exercise

### 8. Write a program to Read & Print a Matrix.

```
ReadMatrix.py - C:\Users\Administrator\Desktop\ONLINE CLASS\Python MCA\Fin...
File Edit Format Run Options Window Help

#Read and Print a Matrix

R = int(input("Enter the number of rows:"))
C = int(input("Enter the number of columns:"))

# Initialize matrix
matrix = []

print("Enter the entries rowwise:")

# For user input
for i in range(R):          # A for loop for row entries
    a = []
    for j in range(C):      # A for loop for column entries
        a.append(int(input()))
    matrix.append(a)

# For printing the matrix
for i in range(R):
    for j in range(C):
        print(matrix[i][j], end = " ")
    print()
```

```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help

Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\Administrator\Desktop\ONLINE CLASS\Python MCA\Final PPT Exercise Project and Syllabus\Exercise\ReadMatrix.py
Enter the number of rows:3
Enter the number of columns:3
Enter the entries rowwise:
1
2
3
4
5
6
7
8
9
1 2 3
4 5 6
7 8 9
>>>
```



## Programming Exercise

9. Write a program to **Transpose** a matrix.



Result Size: 668 x 476

```
#Transpose of a matrix
```

```
X = [[1,2],  
      [3,4],  
      [5,6]]
```

```
result = [[0,0,0],  
          [0,0,0]]
```

```
for i in range(len(X)):  
    for j in range(len(X[0])):  
        result[j][i] = X[i][j]
```

```
for r in result:  
    print(r)
```

```
[1, 3, 5]  
[2, 4, 6]
```



## Programming Exercise

10. Write a program for **Matrix Multiplication** in Python



Result Size: 668 x 476

```
X = [[1, 2],
      [3, 4],
      [4, 5]]

Y = [[1, 2, 3],
      [4, 5, 6]]

result = [[0, 0, 0],
          [0, 0, 0],
          [0, 0, 0]]

for i in range(len(X)):

    for j in range(len(Y[0])):

        for k in range(len(Y)):
            result[i][j] += X[i][k] * Y[k][j]

for r in result:
    print(r)
```

```
[9, 12, 15]
[19, 26, 33]
[24, 33, 42]
```



## Programming Exercise

### 11. Write a program to print **Factorial** using **Function**



Result Size: 668 x 476

```
#Finding Facorial of a number using Function
def fact(n):
    f = 1
    for i in range(1,n+1):
        f = f * i
    print("The factorial of",n,"is",f)

num = 3
#num = int(input("Enter a number: "))

# check if the number is negative, zero or positive
if num < 0:
    print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    fact(num)
```

The factorial of 3 is 6



## Programming Exercise

### 12. Write a program to print **Fibonacci Sequence** using **Recursion**



Result Size: 668 x 476

```
# Display the Fibonacci Sequence

def fib(n):
    if n <= 1:
        return n
    else:
        return(fib(n-1) + fib(n-2))

terms = 10
#terms = int(input("Enter the terms: "))

# check if the number of terms is valid or not
if terms <= 0:
    print("Plese enter a positive integer")
else:
    print("Fibonacci Sequence:")
    for i in range(terms):
        print(fib(i))
```

Fibonacci Sequence:

0  
1  
1  
2  
3  
5  
8  
13  
21  
34

# Queries ?