PYTHON

1. Conditional Statement

search.py demo1.py

demo1.py > ...

```python
num1= int(input("Enter an number: "))

if num1%2==0:
    print(num1, " is an even number.")
else:
    print(num1, " is an odd number.")

```

```
Enter an number: 4
4  is an even number.
```

```
Enter an number: 5
5  is an odd number.
```

2. Arithmetic Operations

```python
def add(num1, num2):
    # pass
    res= num1+ num2;
    print("Addition of", num1, "and", num2, "is: ", res)

def sub(num1, num2):
    res= num2- num1
    print("Subtraction of", num1, "from", num2, "is: ", res)

def mul(num1, num2):
    res= num1* num2;
    print("Multiplication of", num1, "and", num2, "is: ", res)

def div(num1, num2):
    res= num2 / num1;
    print("Division of", num2, "by", num1, "is: ", res)
```

```python
def main():
    num1 = int(input("Enter first number: "))
    num2 = int(input("Enter second number: "))
    add(num1, num2)
    sub(num1, num2)
    mul(num1, num2)
    div(num1, num2)

if __name__=='__main__':
    main()
```

```
Enter first number: 4
Enter second number: 8
Addition of 4 and 8 is:  12
Subtraction of 4 from 8 is:  4
Multiplication of 4 and 8 is:  32
Division of 8 by 4 is:  2.0
```

3. Break Statement:

```python
for letter in 'Python':
    if letter == 'h':
        break
    print('Current Letter: ', letter)

var=10
while var>0:
    print('Current variable value: ', var)
    var-=1
    if var==5:
        break
else:
    print("Good Bye!")
```

```
Current Letter:  P
Current Letter:  y
Current Letter:  t
Current variable value:  10
Current variable value:  9
Current variable value:  8
Current variable value:  7
Current variable value:  6
```

4. Continue Statement

continue.py > ...

```python
for letter in 'Python':
    if letter =='h':
        continue
    print('Current Letter: ', letter)

var=10
while var>0:
    var-=1
    if var==5:
        continue
    print('Current variable value: ', var)
print("Good Bye!")
```

```
Current Letter:  P
Current Letter:  y
Current Letter:  t
Current Letter:  o
Current Letter:  n
Current variable value:  9
Current variable value:  8
Current variable value:  7
Current variable value:  6
Current variable value:  4
Current variable value:  3
Current variable value:  2
Current variable value:  1
Current variable value:  0
Good Bye!
```

5. Connecting and working with MySQL database:

```python
import mysql.connector

def database():

    database= mysql.connector.connect(
        host="localhost",
        user="root",
        passwd="123456",
        database="amdocsb4"
    )

    cursorObject= database.cursor()

    cursorObject.execute("Create database amdocsb4")

    cursorObject.execute("Create table student (name varchar(255), branch varchar(255), roll int, section varchar(50), age

    sql="Insert into student values(%s, %s, %s, %s, %s)"
    val=("Ram", "CSE", 101, "A", 21)
    cursorObject.execute(sql, val)
    database.commit()

    query= 'Select * from student'
    cursorObject.execute(query)

    myresult= cursorObject.fetchall()

    for x in myresult:
        print(x)

    query="Update student set name='Ram Kapoor' where name='Ram' "
    cursorObject.execute(query)
```

```
33      database.commit()
34
35      cursorObject.close()
36      database.close()
37
38  def main():
39      database()
40
41  if __name__=="__main__":
42      main()
```

```
('Ram Kapoor', 'CSE', 101, 'A', 21)
```

6. Encapsulation

encapsulation.py > Base > __init__

```python
# Python program to
# demonstrate protected members

# Creating a base class
class Base:
    def __init__(self):

        # Protected member
        self._a = 2
        print(self._a)

# Creating a derived class
class Derived(Base):
    def __init__(self):

        # Calling constructor of
        # Base class
        Base.__init__(self)
        print("Calling protected member of base class: ",
              self._a)

        # Modify the protected variable:
        self._a = 3
        print("Calling modified protected member outside class: ",
              self._a)


obj1 = Derived()

obj2 = Base()

# Calling protected member
# Can be accessed but should not be done due to convention
print("Accessing protected member of obj1: ", obj1._a)

# Accessing the protected variable outside
print("Accessing protected member of obj2: ", obj2._a)
```

```
2
Calling protected member of base class:  2
Calling modified protected member outside class:  3
2
Accessing protected member of obj1:  3
Accessing protected member of obj2:  2
```

7. For Else :

forElse.py > ...

```python
for count in range(6):
    print("Iteration no. {}".format(count))
else:
    print("For loop over. Now in else block")
print("End of for loop")
```

```
Iteration no. 0
Iteration no. 1
Iteration no. 2
Iteration no. 3
Iteration no. 4
Iteration no. 5
For loop over. Now in else block
End of for loop
```

8. For loop:

forLoop.py > ...

```python
zen='''
Beautiful is better than ugly.
Explicit is betten than implicit.
Simple is better than complex.
Complex is better than complicated.
'''

for char in zen:
    if char not in 'aeiou':
        print(char, end='') # end is delimeter
```

```
Btfl s bttr thn gly.
Explct s bttn thn mplct.
Smpl s bttr thn cmplx.
Cmplx s bttr thn cmplctd.
```

```
numbers=(34, 54, 67, 21, 78, 97, 45, 44, 80, 19)
total=0
for num in numbers:
    total+=num
print("Total= ", total)
```

```
Total=  539
```

```
# enter one number and print its table

num= int(input("Enter a number: "))
print("Table of ", num, " is: ")
for i in range(10): # range(firstValue, endValue, incrementValue)
    print(num,'*', i+1, '= ',num*(i+1))
```

```
Enter a number: 3
Table of  3  is:
3 * 1 =  3
3 * 2 =  6
3 * 3 =  9
3 * 4 =  12
3 * 5 =  15
3 * 6 =  18
3 * 7 =  21
3 * 8 =  24
3 * 9 =  27
3 * 10 =  30
```

9. Nested If:

nestedIf.py > ...

```python
num=9
print("num= ", num)

if num%2==0:
    if num%3==0:
        print("Divisible by 2 and 3")
    else:
        print("Divisible by 2 not divisible by 3")
else:
    if num%3==0:
        print("Divisible by 3 and not by 2")
    else:
        print("Not divisible by 3 and 2 both ")
```

```
num=  9
Divisible by 3 and not by 2
```

10. Non-parameterized constructor:

non_parameterized.py > ...

```python
class Student:
    #Constructor - non parameterized
    def __init__(self) :
        print("This is a non parameterized constructor")
    def show(self, name):
        print("Hello", name)

student = Student()
student.show("John")
```

```
This is a non parameterized constructor
Hello John
```

11. Parameterized Constructor

paramitarized.py > ...

```python
class Employee:
    def __init__(self, name, id):
        self.id=id
        self.name= name

    def display(self):
        print("ID: %d \nName: %s" % (self.id, self.name))

emp1= Employee("John", 101)
emp2= Employee("David", 102)

# accessing display() method to print employee 1 information
emp1.display()

# accessing display() method to print employee 2 information
emp2.display()
```

```
ID: 101
Name: John
ID: 102
Name: David
```

12. Check for Prime number

prime.py > checkPrime

```python
import math

def checkPrime(n):
    if n <= 1:
        return "Non-Prime"
    if n == 2:
        return "Prime"
    if n%2==0:
        return "Non-Prime"

    for i in range(2, int(math.sqrt(n)) + 1, 2):  # Check only odd divisors
        if n % i == 0:
            return "Non-Prime"

    return "Prime"

def main():
    result = checkPrime(6)
    print(result)
    print(int(math.sqrt(36)))

if __name__ == "__main__":
    main()
```

```
Non-Prime
6
```

13. Private variable

```python
class Base1:
    def __init__(self):
        self.p= "Amdocs" #public
        self.__q="Amdocs" #private
        print("private: ",self.__q)

class Derived1(Base1):
    def __init__(self):
        Base1.__init__(self)
        print("Calling private member of private class: ")
        print(self.__q)

ob1= Base1()

print(ob1.p)
# obj2= Derived1()
```

```
private:  Amdocs
Amdocs
```

14. Range Loop

rangeLoop.py > ...

```python
fact=1
n= int(input("Enter a number: "))

for x in range(1, n+1):
    print(x)
    fact= fact*x
print("Factorial of {} is {}".format(n, fact))
```

```
Enter a number: 4
1
2
3
4
Factorial of 4 is 24
```

15. Split in regex

regex_split.py > ...

```python
import re

string = 'Twelve:12 Eighty nine:89.rdgf7g'
pattern = '\d+'

result= re.split(pattern, string)
print(result)
```

```
['Twelve:', ' Eighty nine:', '.rdgf', 'g']
```

16. Pattern matching in regex

```python
import re

#Our pattern
pattern= "hello"

#Returns a match object if found else Null
match= re.match(pattern, "hello world")

print(match)
print("Span: ", match.span()) #return the tuple(start, end)
print("Start: ", match.start()) #return the starting index
print("End: ", match.end()) #returns the ending index

pattern= '^a...s$'
test_string= 'abyss'
result= re.match(pattern, test_string)

if result:
    print("Search successful")
else:
    print("Search unsuccessful")
```

```
<re.Match object; span=(0, 5), match='hello'>
Span:  (0, 5)
Start:  0
End:  5
Search successful
```

17. Remove whitespaces with regex

removeSpaces.py > ...

```python
import re

string = 'abc 12\ de 23 \n f45 6'

#matches all whitespace characters
pattern= '\s+'

replace= ''

new_string = re.sub(pattern, replace, string)
print(new_string)
```

```
abc12\de23f456
```

18. Search in regex

```python
import re

string= "Python is fun"

#check if 'Python' is at the beginning
match= re.search('\APython', string)

print(match)
if match:
    print("Pattern found inside the string")
else:
    print("Pattern not found")
```

```
<re.Match object; span=(0, 6), match='Python'>
Pattern found inside the string
```

19. Student details

```python
# ask student for name, 5 subject marks, print total marks and percentage

# >75 distinction
# >65 excellent
# >50 very good
# >40 pass
# else fail

def calculateGrade(percentage):
    if(percentage>75):
        return "Distinction"
    elif(percentage>65):
        return "Excellent"
    elif (percentage>50):
        return "Very Good"
    elif (percentage>40):
        return "Pass"
    else:
        return "Fail"


def main():
    name= input("Enter your name: ")
    maths_marks= int(input('Enter your maths marks: '))
    eng_marks= int(input('Enter your english marks: '))
    science_marks= int(input("Enter your science marks: "))
    gk_marks= int(input("Enter your general knowledge marks: "))
    computer_marks =int(input('Enter your computer makrs: '))

    total= maths_marks+eng_marks+science_marks+gk_marks+computer_marks
    percentage= total /5
    print('Hi ', name, ',')
    print('Your total marks is: ', total)
    print('Your percentage is: ', percentage, '%')
    print('Your grade is: ',calculateGrade(percentage))

if __name__ == "__main__":
    main()
```

```
Enter your name: Rajiv Kumar
Enter your maths marks: 77
Enter your english marks: 65
Enter your science marks: 88
Enter your general knowledge marks: 83
Enter your computer makrs: 91
Hi  Rajiv Kumar ,
Your total marks is:  404
Your percentage is:  80.8 %
Your grade is:  Distinction
```

20. Switch case

switchCase.py > ...

```python
def checkVowel(n):
    match n:
        case 'a': return "Vowel alphabet"
        case 'e': return "Vowel alphabet"
        case 'i': return "Vowel alphabet"
        case 'o': return "Vowel alphabet"
        case 'u': return "Vowel alphabet"
        case _: return "Consonant"

print(checkVowel('i'))
print(checkVowel('m'))
```

```
Vowel alphabet
Consonant
```

21. While Loop

```
zen='''
Beautiful is better than ugly.
Explicit is betten than implicit.
Simple is better than complex.
Complex is better than complicated.
'''

i=0
while i<len(zen):
    char= zen[i]
    if char not in 'aeiou':
        print(char, end='')
    i+=1
```

```
Btfl s bttr thn gly.
Explct s bttn thn mplct.
Smpl s bttr thn cmplx.
Cmplx s bttr thn cmplctd.
```

22. Inheritance

```python
class Animal:
    def speak(self):
        print("Animal Speaking")

#child class Dog inherits the base class Animal

class Dog(Animal):
    def bark(self):
        print("Dog Barking")

d= Dog()
d.bark()
d.speak()

a= Animal()
a.speak()
```

```
Dog Barking
Animal Speaking
Animal Speaking
```

23. Multi-level Inheritance

3_inhmulti.py > ...

```python
class Animal:
    def speak(self):
        print("Animal Speaking")

#The child class Dog inherits the base class Animal
class Dog(Animal):
    def bark(self):
        print("Dog Barking")

#The child class Dogchild inherits another child class Dog
class DogChild(Dog):
    def eat(self):
        print("Eating bread: ")

d= DogChild()
d.eat()
d.bark()
d.speak()
```

```
Eating bread:
Dog Barking
Animal Speaking
```

24. Multiple Inheritance

3_inhmultiple.py > ...

```python
class Calculation1:
    def Summation(self, a, b):
        return a+b

class Calculation2:
    def Multiplication(Self, a, b):
        return a*b

class Derived(Calculation1, Calculation2):
    def Divide(self,a,b):
        return a/b

d=Derived()

print(d.Summation(10, 20))
print(d.Multiplication(10, 20))
print(d.Divide(10, 20))
```

```
30
200
0.5
```

25. isInstance :

```python
class Calculation1:
    def Summation(self, a, b):
        return a+b

class Calculation2:
    def Multiplication(Self, a, b):
        return a*b

class Derived(Calculation1, Calculation2):
    def Divide(self,a,b):
        return a/b

d=Derived()
```

```
print(isinstance(d, Derived))
```

```
True
```

26. isSubClass:

```
class Calculation1:
    def Summation(self, a, b):
        return a+b

class Calculation2:
    def Multiplication(Self, a, b):
        return a*b

class Derived(Calculation1, Calculation2):
    def Divide(self,a,b):
        return a/b

d=Derived()

print(issubclass(Derived, Calculation2))
print(issubclass(Calculation1, Calculation2))
```

```
True
False
```

27. Method Overriding:

```python
class Bank:
    def getroi(self):
        return 10

class SBI(Bank):
    def getroi(self):
        return 7

class ICICI(Bank):
    def getroi(self):
        return 8

b1= Bank()
b2= SBI()
b3= ICICI()

print("Bank rate of interest", b1.getroi())
print("SBI rate of interest", b2.getroi())
print("ICII rate of interest", b3.getroi())
```

```
Bank rate of interest 10
SBI rate of interest 7
ICII rate of interest 8
```

28. Polymorphism:

3_poly.py > ...

```python
class xyz():
    def websites(self):
        print("Amdocs is a website out of many available on net")

    def topic(self):
        print("Python is out of many topics about technology on Amdocs")

    def type(self):
        print("Amdocs is an developed website")

class PQR():
    def websites(self):
        print("Amdocs is a website out of many available on net")

    def topic(self):
        print("Python is out of many topics about technology on Amdocs")

    def type(self):
        print("Amdocs is an developed website")

obj_jtp = xyz()
obj_pvl = PQR()
```