PYTHON

CODES

1. check whether the given year is leap year or not

Here is a simple Python code snippet to check if a year is a leap year:

```
def is_leap(year):
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                return True
            else:
                return False
        else:
            return True
    else:
        return False

year = int(input("Enter a year: "))
if is_leap(year):
    print(year, "is a leap year")
else:
    print(year, "is not a leap year")
```

2. Conditions for selecting a model

```
height=float(input())
weight=float(input())
gender=input()
if height>5.8:
    if weight<50:
        if gender=="Female":
            print("Selected")
```

```
        else:
            print("Not selected")
    else:
        print("Not selected")
else:
    print("Not selected")
```

3. Check whether the given number is prime number or not without flag

```
n=int(input())
for i in range(2,n):
    if n%i==0:
       print("Not prime")
       break
else:
    print("PRIME")
```

4. Check whether the given number is prime number or not with flag

```
n=int(input())
flag=0
for i in range(2,n//2):
    if n%i==0:
        flag=1
        break
if flag==1:
    print("Not prime")
else:
    print("PRIME")
```

5. GCD

```
a=int(input())
b=int(input())
m=min(a,b)
```

```
g=1
for i in range(1,m+1):
    if a%i==0 and b%i==0:
        g=i
print(g)
```

6. LCM

```
a=int(input())
b=int(input())
m=max(a,b)
g=m
for i in range(m,(a*b)+1):
    if i%a==0 and i%b==0:
        g=i
        break
print(g,"is the LCM")
```

7. Reverse of a number

```
n=int(input())
rev=0
while n!=0:  (OR) while n>0:
    rem=n%10
    rev=rev*10+rem
    n=n//10
print(rev)
```

8. Sum of all the digits in the given number

```
n=int(input())
sum=0
while n>0:
    rem=n%10
    sum=sum+rem
```

```
    n=n//10
print(sum)
```

9. Multiplication of all the digits in the given number

```
n=int(input())
mul=1
while n>0:
    rem=n%10
    mul=mul*rem
    n=n//10
print(mul)
```

10. Check whether the given number is palindrome or not!!!

```
n=int(input())
m=n
temp=0
while n>0:
    rem=n%10
    temp=temp*10+rem
    n=n//10
print(temp)
if m==temp:
    print("Palindrome number")
else:
    print("not a palindrome")
```

11. Count the number of digits in a number

```
n=int(input())
r=0
while n!=0:
    r=r+1
```

```
    n=n//10
print(r)
```

12. Count the number of digits in a number is odd or even

```
n=int(input())
even=0
odd=0
while n!=0: (OR) while n>0:
    rem=n%10
    if rem%2==0:
        even=even+1
    else:
        odd=odd+1
    n=n//10
print("Even numbers=",even)
print("odd numbers=",odd)
```

13. Factorial of a number

```
n=int(input())
fact=1
for i in range(1,n+1):
    if n!=0:
         fact=fact*i
print(fact)
```

14. Factorial of a number in reverse

```
n=int(input())
fact=1
for i in range(n,0,-1):
    if n!=0:
```

```
          fact=fact*i
print(fact)
```

15. Fibonacci series

```
n=int(input())
a=0
b=1
print(a,b,end="  ")
for i in range(3,n+1):
   c=a+b
   print(c,end="  ")
   a=b
   b=c
```

16. Matrix Pattern program

```
n=int(input())
for i in range(0,n):
    for j in range(1,n+1):
        print(j,end="  ")
    print()
```

17. Matrix Pattern program for odd and even

```
n=int(input())
for i in range(0,n):
    for j in range(1,n+1):
        print(j*2-1,end="  ") #for even print(j*2,end="  ")
    print()
```

18. Matrix Pattern program

```
n=int(input())
for i in range(1,n+1):
    for j in range(i,n+i):
        print(j,end="  ")
    print()
```

19. Matrix Pattern program from characters

```
n=int(input())
for i in range(0,n):
    for j in range(i,n+i):
        print(chr(65+j),end="  ")
    print()
```

20. Pattern program

```
n=int(input())
for i in range(1,n+1):
    for j in range(1,i+1):
        print(j,end="  ")
    print()
```

21. Pattern program

```
n=int(input())
for i in range(1,n+1):
    for j in range(i,i+i):
        print(j,end="  ")
    print()
```

22. Pattern program

```
n=int(input())
for i in range(1,n+1):
    for j in range(i,i+i):
        print(j*2-1,end="  ")
    print()
```

23. Pattern program

```
n=int(input())
for i in range(n,0,-1):
    for j in range(i,0,-1):
        print(j,end="  ")
    print()
```

24. Pattern program

```
n=int(input())
for i in range(n):
    for j in range(n,0+i,-1):
        print(j,end="  ")
    print()
```

25. pattern program(*s)

```
n=int(input())
for i in range(n):
    for j in range(n):
        print("*",end="  ")
    print()
```

26. pattern program(*s) Single border

```
n=int(input())
for i in range(n):
```

```
    if i==0 or i==n-1:
        for j in range(n):
            print("*",end=" ")
        print()
    else:
        for j in range(n):
            if j==0 or j==n-1:
                print("*",end=" ")
            else:
                print(" ",end=" ")
        print()
```

27. pattern program(*s) Double border

```
n=int(input())
for i in range(n):
    if i==0 or i==1 or i==n-1 or i==n-2:
        for j in range(n):
            print("*",end=" ")
        print()
    else:
        for j in range(n):
            if j==0 or j==1 or j==n-1 or j==n-2:
                print("*",end=" ")
            else:
                print(" ",end=" ")
        print()
```

28. pattern program(*s) Single border with a middle star

```
n=int(input())
for i in range(n):
    if i==0 or i==n-1:
        for j in range(n):
```

```
            print("*",end=" ")
        print()
    elif i==n//2:
            for j in range(n):
                 if j==0 or j==n//2 or j==n-1:
                    print("*",end=" ")
                 else:
                    print(" ",end=" ")
            print()
    else:
        for j in range(n):
            if j==0 or j==n-1:
                print("*",end=" ")

            else:
                print(" ",end=" ")
        print()
```

29. pyramid pattern

```
n=int(input())
for i in range(1,n+1):
    for j in range(1,n-i+1):
        print(" ",end=" ")
    for j in range(0,2*i-1):
        print("*",end=" ")
    print()
```

30. pyramid pattern single border

```
n=int(input())
for i in range(1,n+1):
    if i==1 or i==n:
```

```
        for j in range(1,n-i+1):
            print(" ",end=" ")
        for j in range(0,2*i-1):
                print("*",end=" ")
        print()
    else:
        for j in range(1,n-i+1):
            print(" ",end=" ")
        for j in range(0,2*i-1):
          if j==0 or j==2*i-2:
              print("*",end=" ")
          else:
            print(" ",end=" ")
        print()
```

31. pyramid pattern up and down

```
n=int(input())
for i in range(1,n+1):
        for j in range(1,n-i+1):
            print(" ",end=" ")
        for j in range(0,2*i-1):
            print("*",end=" ")
        print()
for i in range(n-1,0,-1):
        for j in range(1,n-i+1):
            print(" ",end=" ")
        for j in range(0,2*i-1):
            print("*",end=" ")
        print()
```

32. Hollow Diamond Pattern Program

```
n=int(input())
for i in range(1,n+1):
```

```
    if i==1:
        for j in range(1,n-i+1):
            print(" ",end=" ")
        for j in range(0,2*i-1):
            print("*",end=" ")
        print()
    else:
        for j in range(1,n-i+1):
            print(" ",end=" ")
        for j in range(0,2*i-1):
            if j==0 or j==2*i-2:
                print("*",end=" ")
            else:
                print(" ",end=" ")
        print()
for i in range(n-1,0,-1):
    if i==1:
        for j in range(1,n-i+1):
            print(" ",end=" ")
        for j in range(0,2*i-1):
            print("*",end=" ")
        print()
    else:
        for j in range(1,n-i+1):
            print(" ",end=" ")
        for j in range(0,2*i-1):
            if j==0 or j==2*i-2:
                print("*",end=" ")
            else:
                print(" ",end=" ")
        print()
```

33. double pattern program

```
n=int(input())
for i in range(1,n+1):
    for j in range(1,n-i+1):
          print(" ",end=" ")
    for j in range (0,2*i-1):
          print("*",end=" ")
    for j in range(1,(n-i+1)*2):
        if j!=(n-i+1)*2-1:
          print(" ",end=" ")
    for j in range (0,2*i-1):
          print("*",end=" ")
    print()
```

34. pattern program "A”

```
n=int(input())
for i in range(1,2*n+1):
    if i==1 or i==(2*n)//2:
        for j in range(n+1):
            print("*",end=" ")
        print()
    else:
        for j in range(n+1):
            if j==0 or j==n:
                print("*",end=" ")
            else:
                print(" ",end=" ")
        print()
```

List

1. List

```
l=[1,2,3,4,5]
print(l[::-1])
Reversing List
```

2. Reversing List

```
l=[1,2,3,4,5]
#print(l[::-1])
for i in range(len(l)-1,-1,-1):
    print(l[i])
```

3. Finding largest element in using List

```
l=[5,8,9,10,3,13,19]
m=l[0]
for i in l:
    if i>m:
        m=i
print(m)
```

4. Finding smallest element

```
l=[5,8,9,10,3,13,19]
m=l[0]
for i in l:
    if i<m:
        m=i
print(m)
```

5. Ascending order using List

```
l=[3,5,9,4,10,2]
p=set(l)
q=list(p)
```

```
print(p)
```

6. Finding 2nd largest using List

```
l=[3,5,9,4,10,2]
p=set(l)
q=list(p)
print(q[-2])
```

7. Finding 2nd smallest using List

```
l=[3,5,9,4,10,2]
p=set(l)
q=list(p)
print(q[1])
```

8. Two Sum

```
nums=[1,2,3,5,9]
target=8
def twosum(nums,target):
    a=[]
    for i in range(len(nums)):
        for j in range(i+1,len(nums)):
            if nums[i]+nums[j]==target:
                a.append(i)
                a.append(j)
    return(a)
    print(twosum(numa,target))
```

9. Three Sum

```
nums=[1,2,3,5,9]
target=17
def sumthree(nums,target):
    a=[]
    for i in range(len(nums)):
        for j in range(i+1,len(nums)):
            for k in range(j+1,len(nums)):
                if nums[i]+nums[j]+nums[k]==target:
                    a.append(i)
                    a.append(j)
                    a.append(k)
    return(a)
print(sumthree(nums,target))
```

10. Removing duplicate in a List

```
l=[1,1,1,2,2,3]
room=[]
for i in l:
    if i not in room:
        room.append(i)
print(room)
```

11. Counting the no of duplicates in a List

```
s=[1,5,2,3,1,4,2,1,3]
p=set(s)
for i in p:
    c=0
    for j in s:
        if i==j:
            c+=1
    print(i,"-
```

12. Sum of all even and odd numbers in the List

```
l=[1,2,3,4,5,6,7,8]
even=0
odd=0
for i in l:
    if i%2==0:
        even+=i
    else:
        odd+=i
print(even)
print(odd)
```

13. Intersection of elements in a List

```
l1=[1,2,3,4,5]
l2=[4,5,6,7,8,9]
l=[]
for i in l1:
    if i in l2:
        l.append(i)
print(l)
```

14. Intersection of non-common element in a List

```
l1=[1,2,3,4,5]
l2=[4,5,6,7,8,9]
l=[]
for i in l1:
    if i not in l2:
        l.append(i)
for i in l2:
    if i not in l1:
        l.append(i)
print(l)
```

15. Find the total prime numbers in a given range in List

```
l1=[1,2,3,4,5]
l2=[4,5,6,7,8,9]
l=[]
for i in l1:
    if i not in l2:
        l.append(i)
for i in l2:
    if i not in l1:
        l.append(i)
print(l)
```

16. Removing duplicate in Strings

```
s="aaabbbcccd"
room=""
for i in s:
    if i not in room:
        room=room+i
print(room)
```

17. Counting the no of duplicates in a Strings

```
s="ababchdaod"
p=set(s)
for i in p:
    c=0
    for j in s:
        if i==j:
            c+=1
    print(i,"-",c)
```

18. Convert a Decimal number into Binary

```
n=int(input())
s=""
while n>0:
    rem=n%2
    s=s+str(rem)
    n=n//2
print(s[::-1])
```

19. Check if a number is a perfect square

20. Check if a number is a power of 2

```
i=1
while 2**i<n:
    i+=1
if 2**i==n:
    print("power of 2")
else:
    print("non power of 2")
```

21. Check if a number is a Armstrong number

```
n=int(input())
d=len(str(n))
a=0
t=n
while t>0:
    rem=t%10
    a=a+(rem**d)
    t=t//10
if a==n:
    print("armstrong")
else:
    print("not an armstrong")
```

22. Check if a given number is a perfect number

```
n=int(input())
r=0
i=1
for i in range(1,n):
    if n%i==0:
        r=r+1
if r==n:
    print("it is a perfect number")
else:
    print("not a perfect number")
```

Slicing

23. Slicing of even and odd

```
a=[1,2,3,4,5,6,7,8,9,10]
even=a[1::2]
odd=a[-2::-2]
l=[]
l.extend(even)
l.extend(odd)
print(l)
```

24. Slicing of even and odd

```
a=[1,2,3,4,5,6,7,8,9,10]
l=a[1::2]+a[-2::-2]
```

```
print(l)
```

25. Slicing of 3,6,9,2,4,6,8,10

```
a=[1,2,3,4,5,6,7,8,9,10]
l=a[2::3]+a[1::2]
print(l)
```

26. Left rotation of list or String

```
a=[1,2,3,4,5]
n=2  #if we change value of n answer will be changed
print(a[-n::]+a[0:-n])


a=[1,2,3,4,5]
n=3
print(a[-n::]+a[0:-n])
```

27. Left rotation of list or String

```
n=int(input())
a=[1,2,3,4,5]
if n>len(a):
     n=n%len(a)
print(a[-n::]+a[0:-n])
```

28. Right rotation of List or String

```
n=int(input())
a=[1,2,3,4,5]
if n>len(a):
     n=n%len(a)
print(a[n::]+a[0:n])
```

29. Check whether the given no is valid phone no

```
n=str(input())
if (len(n))==10:
    if n[0]=='6' or n[0]=='7'  or n[0]=='8'  or n[0]=='9':
        if n.isdigit():
             print("valid")
             exit
        else:
            print("invalid phone no")
    else:
         print("invalid")
else:
    print("invalid")
```

30. Reversing the string using split and join

```
s="hey how are you"
output="you are how hey"
l=list(s.split(" "))
l=l[::-1]
ans=" ".join(l)
print(ans)
```

31. Reversing each element in string using split and join

```
s="hey how are you"
l=list(s.split(" "))
s=""
for i in l:
    s=s+i[::-1]+" "
print(s)
```

32. Checking if a string is a palindrome or not

```
n="Madam"
n1=n.upper()
temp=0
temp=n1[::-1]
if n1==temp:
    print("It is palindrome")
else:
    print("not a palindrome")
```

33. Reverse the string(god)

```
n="dog"
print(n[::-1])
```

34. Anagram

```
a=input()
b=input()
s=set(a)
s2=set(b)
if s==s2:
    for i in s:
        if a.count(i)!=b.count(i):
            print("not anagram")
            break
        else:
          print("Anagram")
else:
    print("not anagram")
```

35.

```
strs=["flower","flow","flight"]
ans=""
s="flower"
flag=0
q=0
for i in s:
        for j in strs:
                if i!=j[q]:
                       print(ans)
                       exit()
        else:
                ans=ans+i
                q+=1
```

36. Longest common prefix

```
class Solution:
    def romanToInt(self, s: str) -> int:
        sum=0
        i=0
        while i<len(s):
            if s[i]=="I":
                if i<=len(s)-2 and s[i+1]=="V":
                    sum=sum+4
                    i=i+2
                elif i<=len(s)-2 and s[i+1]=="X":
                    sum=sum+9
                    i=i+2
                else:
                    sum=sum+1
                    i=i+1
                elif s[i]=="V":
                    sum=sum+5
                    i=i+1
                elif s[i]=="x":
```

```
                    if i<=len(s)-2 and s[i+1]=="L":
                        sum=sum+40
                        i=i+2
                    elif i<=len(s)-2 and s[i+1]=="C":
                        sum=sum+90
                        i=i+2
                    else:
                        sum=sum+
```

Recursion

37. Recursion

```
def recur():
    print("hey")
    recur()
recur()
```

38. Recursion runs default 999 times

```
def recur(x):
    print(x)
    x+=1
    recur(x)
recur(1)
```

39. Recursion factorial (decreasing)

```
def fact(x):
    if x==1:
        return x
    return x*fact(x-1)
n=int(input())
```

```
a=fact(n)
print(a)
```

40. Recursion factorial (increasing)

```
def fact(a,x):
    if a==x:
       return x
    return a*fact(a+1,x)
n=int(input())
a=fact(1,n)
print(a)
```

41. Fibonacci series using Recursion

```
def fib(x):
    if x<=1:
       return x
    return fib(x-1)+fib(x-2)
n=int(input())
a=fib(n)
print(a)
```

42. Recursion Fibonacci series(reverse of no)

```
def rev(x,res):
    if x<=0:
       return res
    rem=x%10
    res=res*10+rem
    return rev(x//10,res)
n=int(input())
res=0
```

```
a=rev(n,res)
print(a)
```

43. Check whether a no is power of 3 using Recursion

```
def power(i,x):
    if 3**i==x:
       return True
    elif 3**i>=x:
       return False
    else:
       return power(i+1,x)
n=int(input())
a=power(1,n)
print(a)
```

44. Find first palindromic string in array

```
class Solution:
    def firstPalindrome(self, words: List[str]) -> str:
        for i in words:
            if i==i[::-1]:
                return i
        else:
            return ""
```

45. Creating a class program using polymorphism

```
class person:
    def __init__(self,x,y,z):
        self.nickname=x
        self.roll=y
        self.height=z
    def run(self):
        print("i can run",self.nickname,self.roll)
```

```
harsha=person("chintu",78,6)
anjali=person("anju,65,5
harsha.run()
anjali.run(
```

46. Abstract method and Class

```
class mobile:
    def functions(self):
        pass
class iphone(mobile):
    def functions(self):
        print("Hey!! i am IPHONE")
class samsung(mobile):
    def functions(self):
        print("Hey!! i am SAMSUNG")
iphone13=iphone()
iphone13.functions()
samsungm31s=samsung()
samsungm31s.functions()
```

47. Encapsulation

```
class car:
    _engine="v8"
    _wires="blue"
    def getter(self):
        print(self._engine)
        print(self._wires)
    def setter(self,engine,wires):
        self._engine=engine
        self._wires=wires
bmw=car()
bmw.setter("v9","red")
bmw.getter()
```

48. Single Level Inheritance

```
class parents:
    def coolness(self):
        print("parents are cool")
class child(parents):
            def coding(self):
              print("i know coding")
rahul=child()
rahul.coolness()
rahul.coding()
```

49. Multilevel Inheritance

```
class parents:
   def coolness(self):
      print("parents are cool")
class child(parents):
   def coding(self):
      print("i know coding")
class child2(child):
   def singing(self):
      print("i can sing")
rahul=child2()
rahul.coolness()
rahul.coding()
rahul.singing()
```

50. Multiple Inheritance

```
class dad:
   def coolness(self):
      print("parents are cool")
class mom:
   def coding(self):
```

```
      print("i know coding")
class child2(dad,mom):
   def singing(self):
      print("i can sing")
rahul=child2()
rahul.coolness()
rahul.coding()
rahul.singing()
```

51. Hierchical Inheritance

```
class parents:
   def coolness(self):
      print("parents are cool")
class child1(parents):
   def coding(self):
      print("i know coding")
class child2(parents):
   def singing(self):
      print("i can sing")
rahul=child1()
bhavana=child2()
rahul.coolness()
bhavana.singing()
rahul.coding()
```

52. Hybrid Inheritance

```
class parents:
    def coolness(self):
        print("parents are cool")
class child(parents):
            def coding(self):
              print("i know coding")
class child2(parents):
```

```
      def dancing(self):
            print("i know dancing")
class child3(child,child2):
      def singing(self):
            print("I can sing")
sahethi=child3()
sahethi.coolness()
sahethi.coding()
sahethi.dancing()
sahethi.singing()
```

53. Overriding

```
class addition:
    def add(self,x,y):
        print(x+y)
class child(addition):
    def add(self,x,y,z):
        print(x+y+z)
i=child()
i.add(5,6,7)
```

54. Roman to integer

```
class Solution(object):
    def romanToInt(self, s):
        sum=0
        i=0
        while i<len(s):
            if s[i]=="I":
                if i<=len(s)-2 and s[i+1]=="V":
                    sum=sum+4
                    i=i+2
                elif i<=len(s)-2 and s[i+1]=="X":
                    sum=sum+9
```

```
                i=i+2
            else:
                sum=sum+1
                i=i+1
        elif s[i]=="V":
            sum=sum+5
            i=i+1
        elif s[i]=="X":
            if i<=len(s)-2 and s[i+1]=="L":
                sum=sum+40
                i=i+2
            elif i<=len(s)-2 and s[i+1]=="C":
                sum=sum+90
                i=i+2
            else:
                sum=sum+10
                i=i+1
        elif s[i]=="L":
            sum=sum+50
            i=i+1
        elif s[i]=="C":
            if i<=len(s)-2 and s[i+1]=="D":
                sum=sum+400
                i=i+2
            elif i<=len(s)-2 and s[i+1]=="M":
                sum=sum+900
                i=i+2
            else:
                sum=sum+100
                i=i+1
        elif s[i]=="D":
            sum=sum+500
            i=i+1
        elif s[i]=="M":
            sum=sum+1000
```

```
                i=i+1
        return sum
```

55. Maximum nesting depth of parenthesis

```
class Solution:
    def maxDepth(self, s: str) -> int:
        c=0
        max=0
        for i in s:
            if i=="(":
               c+=1
            if max<c:
               max=c
            if i==")":
               c-=1
        return max
```