Sommario....1

introduzione alla programmazione a oggetti

Java

```
public class esercizio49
{
   public static void main(String[] args)
   {
dataClass dataObj = new dataClass();
dataObj.giorno=9;
dataObj.mese=1;
dataObj.anno=2021;
      System.out.println(dataObj.giorno+"-"+dataObj.mese+"-"+dataObj.anno);
   }
}
public class dataClass
{
   public int giorno;
   public int mese;
   public int anno;
}
```

Python

```
class dataClass:
    giorno=0
    mese=0
    anno=0
    def data(self):
        print(str(dataObj.giorno)+"-"+str(dataObj.mese)+"-"+str(dataObj.anno))
dataObj = dataClass()
dataObj.giorno=9
dataObj.mese=1
dataObj.anno=2021
dataObj.data()
```

Try Catch Except

Java

```
class leggiNumero
{
public static void main(String[] args)
{
InputStreamReader input = new InputStreamReader(System.in);
BufferedReader tastiera = new BufferedReader(input);
String str;
int num;
try
{
str=tastiera.readLine();
num=Integer.valueOf(str).intValue();
System.out.println("Questo e' il numero letto: "+num);
}
catch(Exception e)
{
System.out.println("\n Numero non corretto!");
return;
}
}
}
```

Python

```
class leggiNumero:
        def __init__(self):
                try:
                        num = int(input("inserisci un numero: "))
                        print("Questo è il numero letto: ",str(num))
                except Exception as e:
                        print("Numero non corretto")
leggiNumero()
```

Ereditarietà

C++

```
#include<iostream>
#include<cstdlib>
using namespace std;
class CerchioClasse
{
 private:
 float raggio;
 public:
 CerchioClasse(float raggio) { this->raggio=raggio; };
 void SetRaggio(float raggio) { this->raggio=raggio; };
 float Area() { return(raggio*raggio*3.14); };
 float Circonferenza() { return(raggio*2*3.14); };
};
class CilindroClasse : public CerchioClasse
{
 private:
 float altezza;
 public:
 CilindroClasse(float raggio,float altezza) : CerchioClasse(raggio)
 {
this->altezza=altezza;
 };
 void SetAltezza(float altezza) { this->altezza=altezza; };
 float Volume() { return (CerchioClasse::Area()*altezza); };
 float Area();
};
float CilindroClasse::Area()
{
 float AreaBase,AreaLaterale;
 AreaBase=CerchioClasse::Area()*2;
AreaLaterale=CerchioClasse::Circonferenza()*altezza;
Circonferenza() della classe Cerchio
 return (AreaBase+AreaLaterale);
};
int main()
{
 CilindroClasse CilindroObj(4,10);
 cout<<"L'area del cilindro e': "<<CilindroObj.Area()<<endl;
 cout<<"Il volume del cilindro e': "<<CilindroObj.Volume()<<endl;
 system("pause");
}
```

Java

```
public class CerchioClasse
{
   private double raggio;
```

```
   public CerchioClasse(double raggio)
   {
      this.raggio=raggio;
   }
   public void setRaggio(double raggio)
   {
      this.raggio=raggio;
   }
   public double area()
   {
      return(raggio*raggio*3.14);
   }
   public double circonferenza()
   {
      return(raggio*2*3.14);
   }
}
public class CilindroClasse extends CerchioClasse
{
   private double altezza;
   public CilindroClasse(double raggio, double altezza)
   {
      super(raggio);
      this.altezza=altezza;
   }
   public void setAltezza(double altezza)
   {
      this.altezza=altezza;
   }
   public double volume()
   {
      return area()*altezza;
   }
}
public class es52
{
   public static void main(String args[])
   {
      CilindroClasse cilindroObj=new CilindroClasse(4.0,10.0);
      System.out.println("area di base: "+ cilindroObj.area());
      System.out.println("volume: " + cilindroObj.volume());
}
}
```

Python

```
class Cerchio:

   def __init__(self, raggio):

self.raggio=raggio

   def area(self):

      return self.raggio*self.raggio*3.14
```

```
    def circonferenza(self):
      return self.raggio*2*3.14
class Cilindro(Cerchio):
    def __init__(self, raggio, altezza):
        super().__init__(raggio)
self.altezza=altezza
    def volume(self):
        return super().area()*self.altezza
cilindroObj = Cilindro(4.0,10.0)
print("area di base = "+str(format(cilindroObj.area(), ".2f")))
print("volume = "+str(format(cilindroObj.volume(), ".2f")))
```

esempio 2 dell'ereditarietà:

```
class PaninoConHamburger:
    def __init__(self):
self.ingredienti = ["Pane","Hamburger"]
    def aggiungiIngrediente(self,ingredienteNuovo):
        pass
    def listaIngredienti(self):
        lista = f"""
        Ingredienti: {", ".join(self.ingredienti)}
        """#join prende tutti gli elementi di una lista e li fa diventare una stringa
        return lista
class BigMac(PaninoConHamburger):
    def __init__(self):
        super().__init__()
    def aggiungiIngrediente(self,ingredienteNuovo):
        if ingredienteNuovo not in self.ingredienti:
self.ingredienti.append(ingredienteNuovo)
pannino = BigMac()
pannino.aggiungiIngrediente("Lattuga, cipolla, formaggio, salsa segreta")
print(pannino.listaIngredienti())
```

Astrazione dell'overloading
(tipo del parametro e numero di parametri)
con Python

Esempio 1: (Tupla)

```
class somma:
        def calcola(self, *args):
                self.result = 0
                for self.x in args:
                        self.result += self.x
                return self.result
s = somma()
print(s.calcola(1,2,3))
print(s.calcola(1,2,3,5,6,8))
```

esempio 2: (Dictionary)

```
class pizzeria():
        def info_pizza(self, **pizza):
                for self.key, self.value in pizza.items():
                print("{0} = {1}".format(key, value))
p = pizzeria()
p.info_pizza(nome="margherita", prezzo=5, ingredienti=["pomodoro", "mozzarella"])
```

Overriding
Java

```
class cerchio
 {
 private double Raggio;
 public cerchio(double Raggio)
 {
this.Raggio=Raggio;
 }
 public void setRaggio (double Raggio)
 {
```

```
this.Raggio=Raggio;
 } public double area()
 {
 return(Raggio*Raggio)*Math.PI;
 }
 public double circonferenza()
 {
 return(2*Raggio*Math.PI);
 }
}
class cilindro extends cerchio
{
 private double altezza;
 public cilindro (double Raggio, double altezza)
{ super(Raggio); this.altezza=altezza; }
 public void setaltezza (double altezza)
{ this.altezza=altezza; }
 public double volume()
{ return(super.area()*altezza); }
 public double area()
 {
 double AreaBase, AreaLaterale;
 AreaBase=super.area()*2;
 AreaLaterale=circonferenza()*altezza;
 return(AreaBase+AreaLaterale);
 }
 }
class main
{
 public static void main(String args[])
 {
 cilindro cilindroObj=new cilindro(4,10);
 System.out.println("L'area del cilindro e': " +cilindroObj.area());
 System.out.println("Il volume e': " +cilindroObj.volume());
 cerchio cerchioObj=new cerchio(2);
```

```
 System.out.println("L'area del cerchio e': "+cerchioObj.area());
 } }
```

Python

```
class Cerchio:
    def __init__(self, raggio):
self.raggio=raggio
    def area(self):
        return self.raggio*self.raggio*3.14
    def circonferenza(self):
      return self.raggio*2*3.14
class Cilindro(Cerchio):
    def __init__(self, raggio, altezza):
        super().__init__(raggio)
self.altezza=altezza
    def area(self):
        areaBase=super().area()*2
        areaLaterale=super().circonferenza()*self.altezza
        return areaBase+areaLaterale
    def volume(self):
        return super().area()*self.altezza
cilindroObj = Cilindro(4,10)
print("L'area del cilindro è "+str(format(cilindroObj.area(), ".2f")))
print("Il volume del cilindro è "+str(format(cilindroObj.volume(), ".2f")))
cerchioObj = Cerchio(2)
```

Classi Astratte

C++

```
#include<iostream>
#include<cstdlib>
using namespace std;
class CerchioClasse
```

```
{
 private:
 float raggio;
 public:
 CerchioClasse(float raggio) { this->raggio=raggio; };
 void SetRaggio(float raggio) { this->raggio=raggio; };
 float Area() { return(raggio*raggio*3.14); };
 float Circonferenza() { return(raggio*2*3.14); };
};
class CilindroClasse : public CerchioClasse
{
 private:
 float altezza;
 public:
 CilindroClasse(float raggio,float altezza) : CerchioClasse(raggio)
 {
this->altezza=altezza;
 };
 void SetAltezza(float altezza) { this->altezza=altezza; };
 float Volume() { return (CerchioClasse::Area()*altezza); };
 float Area();
};
float CilindroClasse::Area()
{
 float AreaBase,AreaLaterale;
 AreaBase=CerchioClasse::Area()*2;
AreaLaterale=CerchioClasse::Circonferenza()*altezza;
Circonferenza() della classe Cerchio
 return (AreaBase+AreaLaterale);
};
int main()
{
 CilindroClasse CilindroObj(4,10);
 cout<<"L'area del cilindro e': "<<CilindroObj.Area()<<endl;
 cout<<"Il volume del cilindro e': "<<CilindroObj.Volume()<<endl;
 system("pause");
}
```

Java

```
public class CerchioClasse
{
   private double raggio;
   public CerchioClasse(double raggio)
   {
      this.raggio=raggio;
   }
   public void setRaggio(double raggio)
   {
      this.raggio=raggio;
   }
   public double area()
   {
```

```
        return(raggio*raggio*3.14);
    }
    public double circonferenza()
    {
        return(raggio*2*3.14);
    }
}
public class CilindroClasse extends CerchioClasse
{
    private double altezza;
    public CilindroClasse(double raggio, double altezza)
    {
        super(raggio);
        this.altezza=altezza;
    }
    public void setAltezza(double altezza)
    {
        this.altezza=altezza;
    }
    public double volume()
    {
        return area()*altezza;
    }
}
public class es52
{
    public static void main(String args[])
    {
        CilindroClasse cilindroObj=new CilindroClasse(4.0,10.0);
        System.out.println("area di base: "+ cilindroObj.area());
        System.out.println("volume: " + cilindroObj.volume());
}
}
```

Python

```
from abc import ABC, abstractmethod

class FiguraGeometrica(ABC):

    def __init__(self):

self.nomeFigura="noName"

    @abstractmethod

    def area(self):

        pass

class Triangolo(FiguraGeometrica):

    def __init__(self):

        super().__init__()

self.base = 0

self.altezza = 0
```

```
    def area(self):
        return(self.base*self.altezza)/2


class Quadrato(FiguraGeometrica):
    def __init__(self):
self.lato=0
    def area(self):
     return self.lato*self.lato
class Rettangolo(FiguraGeometrica):
    def __init__(self):
self.base=0
self.altezza=0
    def area(self):
  return self.base*self.altezza
rettangoloObj = Rettangolo()
triangoloObj = Triangolo()
quadratoObj = Quadrato()
rettangoloObj.base=2
rettangoloObj.altezza=17
triangoloObj.base=12
triangoloObj.altezza=7
quadratoObj.lato=10
print(rettangoloObj.base)
print(triangoloObj.nomeFigura)
print(quadratoObj.area())
```

esempio 2 di classi astratte

```
from abc import ABC, abstractmethod
class Vehicle(ABC):
    @abstractmethod
    def numberofwheels(self):
        pass
class Car(Vehicle):
    def numberofwheels(self):
print("A Car has 4 wheels")
```

```
class Bike(Vehicle):
    def numberofwheels(self):
print("A Bike has 2 wheels")
carObj = Car()
carObj.numberofwheels()
bikeObj = Bike()
bikeObj.numberofwheels()
```

Associazione 1 a 1

Java

```
public class Nazione
{
private String nomeNazione;
private Capitale capitaleAssociata;
```

```
public Nazione(String nomeNazione)
{
this.nomeNazione=nomeNazione;
}
public String getNomeNazione()
{
return nomeNazione;
}
public void setNomeNazione(String nomeNazione)
{
this.nomeNazione=nomeNazione;
}
public void setCapitale(Capitale capitaleAssociata)
{
this.capitaleAssociata=capitaleAssociata;
}
public Capitale getCapitale()
{
return capitaleAssociata;
}
}

public class Capitale
{
private String nomeCapitale;
private Nazione nazioneAssociata;
public Capitale(String nomeCapitale)
{
this.nomeCapitale=nomeCapitale;
}
public String getNomeCapitale()
{
return nomeCapitale;
}
public void setNomeCapitale(String nomeCapitale)
{
this.nomeCapitale=nomeCapitale;
}
public void setNazione(Nazione nazioneAssociata)
{
this.nazioneAssociata=nazioneAssociata;
}
public Nazione getNazione()
{
return nazioneAssociata;
}
}
class Main
{
public static void main(String args[])
{
String s="";
```

```
String s1="";
Nazione nazioneObj;
nazioneObj= new Nazione("Italia");
System.out.println ("nome della Nazione: "+nazioneObj.getNomeNazione());
Capitale capitaleObj;
capitaleObj= new Capitale("Roma");
System.out.println ("nome della Capitale: "+capitaleObj.getNomeCapitale());
nazioneObj.setCapitale(capitaleObj);
capitaleObj.setNazione(nazioneObj);
s=s+"Nazione: "+nazioneObj.getNomeNazione()+" ";
s=s+"Capitale: "+nazioneObj.getCapitale().getNomeCapitale();
System.out.println ("dati "+s);
}
}
```

Python

```
class Nazione:

    def __init__(self, nomeNazione):

self.nomeNazione=nomeNazione

    def associaCapitale(self, capitaleAssociata):

self.capitaleAssociata=capitaleAssociata

class Capitale:

    def __init__(self, nomeCapitale):

self.nomeCapitale=nomeCapitale

    def associaNazione(self, nazioneAssociata):

self.nazioneAssociata=nazioneAssociata

nazioneObj = Nazione("Italia")

capitaleObj = Capitale("Roma")

nazioneObj.associaCapitale(capitaleObj)

capitaleObj.associaNazione(nazioneObj)

s = "Nome della Nazione: {} \nNome della Capitale: {}".format(nazioneObj.nomeNazione,
nazioneObj.capitaleAssociata.nomeCapitale)

print(s)
```

Array di oggetti

Java

```
class Persona
{
private String nome;
```

```
private String cognome;
private int eta;
private String colorePreferito;
public Persona(String nome, String cognome, int eta, String colorePreferito)
{
this.nome=nome;
this.cognome=cognome;
this.eta=eta;
this.colorePreferito=colorePreferito;
}
public String getNome()
{
return nome;
}
public String getCognome()
{
return cognome;
}
public int getEta()
{
return eta;
}
public String getColorePreferito()
{
return colorePreferito;
}
}
class Main
{
public static void main(String args[])
{
Persona[] personeObj = new Persona[3];
Persona persObj1 = new Persona("Luca","Rossi",23,"blue");
Persona persObj2 = new Persona("Marco","Verdi",35,"rosso");
Persona persObj3 = new Persona("Anna","Neri",25,"giallo");
Persona[0] = persObj1;
Persona[1] = persObj2;
Persona[2] = persObj3;
System.out.println(Persona[0].getNome());
}
}
```

Python

```
class Persona:
    def __init__(self, nome, cognome, eta, colorePreferito):
self.nome=nome
self.cognome=cognome
        self.eta=eta
self.colorePreferito=colorePreferito
```

```
persone = []
persona1Obj = Persona("Luca","Rossi",23,"blue")
persona2Obj = Persona("Marco","Verdi",35,"rosso")
persona3Obj = Persona("Anna","Neri",25,"giallo")
persone.append(persona1Obj)
persone.append(persona2Obj)
persone.append(persona3Obj)

print(persone[0].nome)
```

esempio 2 di array di oggetti

```
class Car:
    def __init__(self, marca, km):
self.marca=marca
        self.km=km
macchine = []
macchine.append(Car("Porsche",10000))
macchine.append(Car("Fiat",200000))
macchine.append(Car("Ferrari",32897))
print(macchine[1].marca)
```

Associazione 1 a N

Python

```
class Azienda:
    dipendentiAssociati = []
    def __init__(self, nomeAzienda, numDipendenti):
self.nomeAzienda=nomeAzienda
self.numDipendenti=numDipendenti
    def associaDipendenti(self, dipendentiAssociati):
self.dipendentiAssociati=dipendentiAssociati
class Dipendente:
    def __init__(self, nome, ruolo):
self.nome=nome
self.ruolo=ruolo
```

```
    def associaAzienda(self, aziendaAssociata):
self.aziendaAssociata=aziendaAssociata
dipendenti = []
aziendaObj = Azienda("The SeedBook", 5)
dip1Obj = Dipendente("Marco", "Manager")
dip2Obj = Dipendente("Charlie", "Pentester")
dip3Obj = Dipendente("Alice", "impiegata")
dip4Obj = Dipendente("Luigi", "impiegato")
dip5Obj = Dipendente("Maria", "spazzino")
dipendenti.append(dip1Obj)
dipendenti.append(dip2Obj)
dipendenti.append(dip3Obj)
dipendenti.append(dip4Obj)
dipendenti.append(dip5Obj)
aziendaObj.associaDipendenti(dipendenti)
for dipendente in dipendenti:
dipendente.associaAzienda(aziendaObj)
s = "Nome dell'azienda: {} con {} dipendenti".format(aziendaObj.nomeAzienda, aziendaObj.numDipendenti)
s+= "\nElenco Dipendenti: "
dip=0
for dipendente in dipendenti:
    s+="\n{}: {}".format(aziendaObj.dipendentiAssociati[dip].nome, aziendaObj.dipendentiAssociati[dip].ruolo)
    dip+=1
print(s)
```

Aggregazione

Java

```
public class Carrozzeria
```

```
{
private String colore;
public Carrozzeria(String colore)
{
this.colore=colore;
}
public String getColore()
{
return colore;
}
public String descrCarrozzeria()
{
String s="";
s=colore;
return s;
}
} public class Motore
{
private int numCilindri;
public Motore(int numCilindri)
{
this.numCilindri=numCilindri;
}
public int getNumCilindri()
{
return numCilindri;
}
public String descrMotore()
{
String s="";
s=" "+numCilindri;
return s;
}
public class Automobile
{
private String marca;
private Carrozzeria carrozzeriaAggregata;
private Motore motoreAggregato;
public Automobile(String marca, Carrozzeria carrozzeriaAggregata, Motore motoreAggregato)
{
this.marca=marca;
this.carrozzeriaAggregata=carrozzeriaAggregata;
this.motoreAggregato=motoreAggregato;
}
public String getMarca()
{
return marca;
}
public String descrAuto()
{
String s="";
s=marca+" "+carrozzeriaAggregata.descrCarrozzeria()+" "+motoreAggregato.descrMotore();
```

```
return s;
}
}
class Main
{
public static void main(String args[])
{
Motore motoreObj;
motoreObj= new Motore(4);
Carrozzeria carrozzeriaObj;
carrozzeriaObj= new Carrozzeria("Rossa");
Automobile automobileObj;
automobileObj=new Automobile("Fiat",carrozzeriaObj,motoreObj);
System.out.println ("descrizione automobile "+automobileObj.descrAuto());
}
}
```

Python

```
class Carrozzeria:
    def __init__(self, colore):
self.colore=colore
class Motore:
    def __init__(self, cilindri):
self.cilindri=cilindri
class Automobile:
    def __init__(self, marca, carrozzeriaAggregata, motoreAggregato):
self.marca=marca
self.carrozzeriaAggregata=carrozzeriaAggregata
self.motoreAggregato=motoreAggregato
    def descrAuto(self):
        s=self.marca+" "+self.carrozzeriaAggregata.colore+" "+str(self.motoreAggregato.cilindri)+" cilindri"
        return s
motoreObj = Motore(5)
carrozzeriaObj = Carrozzeria("Rossa")
automobileObj = Automobile("Fiat",carrozzeriaObj,motoreObj)
print("Descrizione automobile: "+automobileObj.descrAuto())
```

esempio 2 di classi aggregate

```
class Motori:
    def __init__(self, numeroMotori):
self.numeroMotori=numeroMotori
```

```
class Ali:
    def __init__(self, aperturaAlare):
self.aperturaAlare=aperturaAlare
class Aereoplano:
    def __init__(self, tipologia, motoriAggregato, aliAggregato):
self.tipologia=tipologia
self.motoriAggregato=motoriAggregato
self.aliAggregato=aliAggregato
    def descrAereo(self):
        return self.tipologia+" con "+str(self.motoriAggregato.numeroMotori)+" motori e un apertura alare di "+str(self.aliAggregato.aperturaAlare)+"m"
motoriObj = Motori(2)
aliObj = Ali(11.4)
aereoplanoObj = Aereoplano("Mig-29",motoriObj,aliObj)
print(aereoplanoObj.descrAereo())
```

GUI con Classi

Java

```
import java.awt.*;
import java.awt.event.*;
public class PulsanteConEvento
{
TextField txtT;
Label lbl;
public PulsanteConEvento()
{
Frame frmF = new Frame("Pulsante con evento");
Label lblL = new Label("inserire il nome: ")
lbl = new Label();
txtT = new TextField(30);
Button btnPulsante = new Button("Saluta");
frmF.setLayout(new FlowLayout());
btnPulsante.addActionListener(new ascoltaPulsante());
frmF.add(lblL);
frmF.add(txtT);
frmF.add(btnPulsante);
frmF.add(lbl);
frmF.pack();
frmF.setSize(600,500);
frmF.setLocation(200,200);
frmF.setVisible(true);
}
```

```
private class ascoltaPulsante implements ActionListener
{
public void actionPerformed(ActionEvent e)
{
  lbl.setText("Buongiorno"+txtT.getText());
}
}
}
```

Python

```
import tkinter as tk

class Saluto:

    #Napoli = "cyan"

    def saluto(self):

self.testo = self.TextField.get()

self.risposta = tk.Label(self.window,text="buongiorno "+self.testo)#,fg="red")

self.risposta.grid(row=2,column=1,sticky="W",pady=5)

self.risposta.after(5000 , lambda: self.risposta.destroy())#Lambda è una funzione senza nome che permette di utilizzare una funzione senza dichiararla

    def __init__(self):

self.window = tk.Tk()

self.window.geometry("500x500")

self.window.title("Saluto")

        #self.window.resizable(False,False)

        #self.window.configure(background=Napoli)#bg=Napoli

self.label = tk.Label(self.window, text="Inseisci il nome:")

self.label.grid(row=0,column=0,sticky="W")

self.TextField = tk.Entry(self.window,width=50)

self.TextField.grid(row=0,column=1,sticky="W",padx=5,pady=5)

self.bottone = tk.Button(self.window,text="Saluta",command=self.saluto)

self.bottone.grid(row=1,column=0,sticky="W",pady=5)

    def start(self):

self.window.mainloop()

finestra = Saluto()

finestra.start()
```

GUI senza Classi

Python

```
import tkinter as tk
```

```
#Napoli = "cyan"
window = tk.Tk()
window.geometry("500x500")
window.title("Saluto")
#window.resizable(False,False)
#window.configure(background=Napoli)
def saluto():
    testo = TextField.get()
    risposta = tk.Label(window,text="buongiorno "+testo)#,fg="red")
risposta.grid(row=2,column=1,sticky="W",pady=5)
risposta.after(5000 , lambda: risposta.destroy())
label = tk.Label(window, text="Inseisci il nome:")
label.grid(row=0,column=0,sticky="W")
TextField = tk.Entry(window,width=50)
TextField.grid(row=0,column=1,sticky="W",padx=5,pady=5)
bottone = tk.Button(window,text="Saluta",command=saluto)
bottone.grid(row=1,column=0,sticky="W",pady=5)
window.mainloop()
```

esempio 2:

```
import tkinter as tk
from tkinter import ttk
window = tk.Tk()
window.geometry("650x650")
window.title("Tutti i Componenti")
etichetta = tk.Label(window, text="Questa è una etichetta")
etichetta.grid(row=0,column=0,sticky="W")
bottone = tk.Button(window,text="Questo è un pulsante")
bottone.grid(row=0,column=1,sticky="W",pady=5)
TextField = tk.Entry(window,width=30)#,show="*")
TextField.insert(0,"Inserire qui il testo")#0 è il primo carattere
TextField.grid(row=1,column=0,sticky="W",padx=5,pady=5)
textArea = tk.Text(window,height=4, width=35)
textArea.insert(1.0,"Inserire in questa area il testo")
textArea.grid(row=3,column=1,sticky="W")
checkBox = tk.Checkbutton(window,text="scelta già spuntata")
```

```
checkBox.select()

checkBox.grid(row=3,column=2,sticky="W")

radio1 = tk.Radiobutton(window,text="Scelta spuntata",value=1)

radio1.select()

radio1.grid(row=4,column=0,sticky="W")

radio2 = tk.Radiobutton(window,text="Scelta non spuntata",value=2)

radio2.grid(row=4,column=1,sticky="W")

combobox = ttk.Combobox(window,values=["Prima Scelta","Seconda
Scelta"])#,state="readonly")#,disabled")

combobox.grid(row=4,column=2,sticky="W",padx=5,pady=5)

combobox.current(0)
```

GUI Materiale Didattico

(Computer)

```
import tkinter as tk

from tkinter import ttk

import os

window = tk.Tk()

#window.iconbitmap('img/logoMatDid.ico')

#window.iconphoto(False, tk.PhotoImage(file='img/logoMatDid.png'))

#False significa che questa immagine icona si applica solo a questa finestra

window.title("Ricerca del Materiale Didatico")

window.geometry("600x600")

window.resizable(False,False)

def cerca():

    url = [

        "http://ennioranucci.altervista.org/materialedidatticoranucci.html",


"http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/Variabili%20costanti%20arra
y%20funzioni%20record.pdf",


"http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/Esercitazioni%20facili%20fac
ili%20FILE%20C++%20Ennio%20Ranucci%2020202021.pdf",


"http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/Esercitazioni%20FILE%20C+
+%20Ennio%20Ranucci%2020192020.pdf",


"http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/OOP%20C++%20Java.pdf",
```

```
"http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/ereditarietaPolimorfismo.pdf",

"http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/astratteVSinterfacce.pdf",
        "http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/java.pdf",

"http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/Esercitazioni%20HTML%20CSS%20Ennio%20Ranucci%2020192020.pdf",

"http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/RaccoltaEsercizi3B20202021.pdf"
      ]
   select = combobox.get()
   if select=="Materiale Didatico":
os.system("start "+url[0])
   elif select=="Classe III - Variabili costanti array funzioni record - C++":
os.system("start "+url[1])
   elif select=="Esercitazioni facili facili FILE C++ Ennio Ranucci 20202021":
os.system("start "+url[2])
   elif select=="Esercitazioni FILE C++ Ennio Ranucci 20192020":
os.system("start "+url[3])
   elif select=="OOP C++ Java Ennio Ranucci 20202021":
os.system("start "+url[4])
   elif select=="Ereditarietà e Polimorfismo":
os.system("start "+url[5])
   elif select=="astratte VS interfacce Ennio Ranucci 20212022":
os.system("start "+url[6])
   elif select=="Esercitazioni Java Ennio Ranucci 20192020":
os.system("start "+url[7])
   elif select=="Esercitazioni HTML CSS Ennio Ranucci 20192020":
os.system("start "+url[8])
   elif select=="Classe III 20202021- Raccolta esercizi a cura di Alessandro Esposito":
os.system("start "+url[9])
label = tk.Label(window, text="Seleziona il materiale didatico interessato:")
label.grid(row=0,column=0,sticky="W")
combobox = ttk.Combobox(window,values=["",
                                "Materiale Didatico",
```

```
                    "Classe III - Variabili costanti array funzioni record - C++",
                    "Esercitazioni facili facili FILE C++ Ennio Ranucci 20202021",
                    "Esercitazioni FILE C++ Ennio Ranucci 20192020",
                    "OOP C++ Java Ennio Ranucci 20202021",
                    "Ereditarietà e Polimorfismo",
                    "astratte VS interfacce Ennio Ranucci 20212022",
                    "Esercitazioni Java Ennio Ranucci 20192020",
                    "Esercitazioni HTML CSS Ennio Ranucci 20192020",
                    "Classe III 20202021- Raccolta esercizi a cura di Alessandro
Esposito"])#,state="readonly")#disabled")
combobox.grid(row=0,column=1,sticky="W",padx=5,pady=5)
combobox.current(0)
combobox.configure(width=55,height=11)
combobox.bind("<<ComboboxSelected>>",lambda select:print("selezionato: "+combobox.get()))
bottone = tk.Button(window,text="Cerca",command=cerca)
bottone.grid(row=1,column=0,sticky="W",pady=5)
window.mainloop()
```

GUI Materiale Didattico

(Android)

```
import tkinter as tk
from tkinter import ttk
import webbrowser as wb
window = tk.Tk()
window.iconbitmap('img/logoMatDid.ico')
window.title("Ricerca del Materiale Didatico")
window.geometry("600x600")
window.resizable(False,False)
def cerca():
    url = [
        "http://ennioranucci.altervista.org/materialedidatticoranucci.html",

"http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/Variabili%20costanti%20arra
y%20funzioni%20record.pdf",

"http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/Esercitazioni%20facili%20fac
ili%20FILE%20C++%20Ennio%20Ranucci%2020202021.pdf",
```

```
"http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/Esercitazioni%20FILE%20C++%20Ennio%20Ranucci%2020192020.pdf",

"http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/OOP%20C++%20Java.pdf",

"http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/ereditarietaPolimorfismo.pdf",

"http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/astratteVSinterfacce.pdf",
        "http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/java.pdf",

"http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/Esercitazioni%20HTML%20CSS%20Ennio%20Ranucci%2020192020.pdf",

"http://ennioranucci.altervista.org/MaterialeDidattico/materialeDidatticoRanucci/RaccoltaEsercizi3B20202021.pdf"
      ]
   select = combobox.get()
   if select=="Materiale Didatico":
wb.open(url[0], new=0, autoraise=True)
   elif select=="Classe III - Variabili costanti array funzioni record - C++":
wb.open(url[1], new=0, autoraise=True)
   elif select=="Esercitazioni facili facili FILE C++ Ennio Ranucci 20202021":
wb.open(url[2], new=0, autoraise=True)
   elif select=="Esercitazioni FILE C++ Ennio Ranucci 20192020":
wb.open(url[3], new=0, autoraise=True)
   elif select=="OOP C++ Java Ennio Ranucci 20202021":
wb.open(url[4], new=0, autoraise=True)
   elif select=="Ereditarietà e Polimorfismo":
wb.open(url[5], new=0, autoraise=True)
   elif select=="astratte VS interfacce Ennio Ranucci 20212022":
wb.open(url[6], new=0, autoraise=True)
   elif select=="Esercitazioni Java Ennio Ranucci 20192020":
wb.open(url[7], new=0, autoraise=True)
   elif select=="Esercitazioni HTML CSS Ennio Ranucci 20192020":
wb.open(url[8], new=0, autoraise=True)
   elif select=="Classe III 20202021- Raccolta esercizi a cura di Alessandro Esposito":
wb.open(url[9], new=0, autoraise=True)
```

```
label = tk.Label(window, text="Seleziona il materiale didatico interessato:")
label.grid(row=0,column=0,sticky="W")
combobox = ttk.Combobox(window,values=["",
                                       "Materiale Didatico",
                                       "Classe III - Variabili costanti array funzioni record - C++",
                                       "Esercitazioni facili facili FILE C++ Ennio Ranucci 20202021",
                                       "Esercitazioni FILE C++ Ennio Ranucci 20192020",
                                       "OOP C++ Java Ennio Ranucci 20202021",
                                       "Ereditarietà e Polimorfismo",
                                       "astratte VS interfacce Ennio Ranucci 20212022",
                                       "Esercitazioni Java Ennio Ranucci 20192020",
                                       "Esercitazioni HTML CSS Ennio Ranucci 20192020",
                                       "Classe III 20202021- Raccolta esercizi a cura di Alessandro
Esposito"])#,state="readonly")#disabled")
combobox.grid(row=0,column=1,sticky="W",padx=5,pady=5)
combobox.current(0)
combobox.configure(width=55,height=11)
combobox.bind("<<ComboboxSelected>>",lambda select:print("selezionato: "+combobox.get()))
bottone = tk.Button(window,text="Cerca",command=cerca)
bottone.grid(row=1,column=0,sticky="W",pady=5)
window.mainloop()
```

File

C++

```
#include <iostream>
#include<fstream>
using namespace std;
fstream numeriFile;
int main()
{
int num, num2;
 num=123;
 numeriFile.open ("numeri.dat", ios::out | ios::binary);
 numeriFile.write((char *) &num, sizeof(num));
 numeriFile.close();
```

```
 numeriFile.open ("numeri.dat", ios::app | ios::binary);
 num=150;
 numeriFile.write((char *) &num, sizeof(num));
 numeriFile.close();
 numeriFile.open ("numeri.dat", ios::in | ios::binary);
 numeriFile.read((char *) &num, sizeof(num));
 cout<<num;
 cout<<endl;
 numeriFile.read((char *) &num2, sizeof(num2));
 cout<<num2;
 numeriFile.close();
 return 0;
}
```

Python

scrittura, append e lettura

```
num=123
numeriFile = open("numeri.txt", "w")
numeriFile.write(str(num)+"#")
numeriFile.close()
numeriFile = open("numeri.txt", "a")
num2=150
numeriFile.write(str(num2))
numeriFile.close()
numeriFile = open("numeri.txt", "r")
strNum=numeriFile.read()
nums = strNum.split("#")
print("Primo numero: "+nums[0])
print("Secondo numero: "+nums[1])
print("Contenuto totale del file: "+strNum)
```

Libreria OS

Python

```
import os
direc = 'fileDir'
os.system('mkdir '+direc)
os.system('cd '+direc)
os.system('type nul > data.txt')
os.getcwd()
f = open('data.txt','w')
f.write("Questo è il contenuto del file")
f.close()
f = open('data.txt', 'r')
contenutoF = f.read()
f.close()
print(contenutoF)
os.getpid()
os.remove('data.txt')
os.system('cd ..')
os.rmdir(direc)
os.system("start http://ennioranucci.altervista.org/materialedidatticoranucci.html")
#os.getgid() #restituisce id utente, ma solo sotto unix
```

Django

Django è un Framework di altissimo livello scritto in python.
Un Framework è un sistema che facilita la creazione di siti
e applicazioni web(aiuta nel campo della sicurezza).

Bootstrap è un Framework utilizzato dal lato grafico
aiuta nel design,ci fornisce Html CSS e JavaScript.
nel loro insieme (Django e Bootstrap) ci consentono di accorciare
i tempi di sviluppo e irrobustire il sito

1) Istallare Django (python -m pip install django)

2) Creazione del progetto (django-admin.py startproject nome)

3) Andare in setting.py e cambiare LANGUAGE_CODE = 'it'

4) Avviare server di sviluppo (python manage.py runserver)

manage.py serve ad aprirlo in un server locale scrivendo localhost:8000

(per conoscere tutti i comandi di manage.py basta digitare: python manage.py)

5) Eseguire le migrazioni di default che servono ad attivare i privilegi di amministratore

tramite questo commando (python manage.py migrate)

6) creare super User(Amministratore) tramite questo comando: python manage.py createsuperuser

inserire username,password e mail(va bene pure vuota)

7)entrare nell'account amministratore tramite(localhost:8000/admin)

8) creare un APP(Una APP in Django è semplicemente un'applicazione riutilizzabile che comprende un insieme di funzionalità utili ad uno scopo ben preciso. Un progetto Djando può essere costituito da 1 o più APP) tramite il comando: python manage.py startapp nome(posts)

9) Aprire la cartella creata e andare su models.py (un modello è una classe Django che rappresenta la tabella di un database con vari campi e datatype serve principalmente a modellare come vogliamo l'APP)

```
from django.db import models

# Create your models here.

class Post(models.Model):

    titolo = models.CharField(max_length=120)

    contenuto = models.TextField()# campo di testo ampio

    data = models.DateTimeField(auto_now=False,auto_now_add=True)#auto_now_add viene settato solo alla creazione l'altro ogni qual volta che viene effettuata una modifica si aggiornerebbe (se avessimo messo True)

    slug = models.SlugField()#SlagField è un etichetta che contiene solo lettere,numeri,_,- ed è un sistema per rendere url user-friendly


def __str__(self):

        return self.titolo
```

10) Una volta creata un APP bisogna istallarla per farlo bisogna andare nel settings.py(Del progetto principale)>>ISTALLED_APPS>>'nome(posts)'

11) Avvertire il database dei nuovi modelli creati tramite le migrazioni. Per prima cosa li dobbiamo trovare(creare) tramite il comando: python manage.py makemigrations

per seconda cosa dobbiamo attivare queste migrazioni create tramite il comando: python manage.py migrate

12) Creare un collegamento tra questa APP con il suo modello e la zona di amministrazione per poter creare dei post. Per fare ciò andiamo nel file admin.py all'interno dell'app e aggiungere il seguente codice

```
from django.contrib import admin

from.models import Post

# Register your models here.

class PostAdmin(admin.ModelAdmin):

    list_display = ["__str__","data"]

    list_filter = ["data"]

search_fields = ["titolo","contenuto"]

    prepopulated_fields = {"slug":("titolo",)}# serve a mettere il slug automatico

class Meta:# fornisce alla PostAdmin informazioni sul modello che sta utilizzando

model = Post

admin.site.register(Post,PostAdmin)
```

13) creare un collegamento tra la pagina principale e la nostra APP. Per fare ciò bisogna aprire il file urls.py(urls.py richiama una views in base alla richiesta) all'interno del progetto principale e scrivere il seguente codice:

```
from django.contrib import admin

from django.urls import path,include

urlpatterns = [

path('admin/', admin.site.urls),

    path('',include('posts.urls'))

]
```

14) Creare gli urls delle varie pagine web utilizzando il file urls.py all'interno della nostra APP e inserire il seguente codice:

```
from django.urls import path

from . import views

from django.views.generic import ListView,DetailView

from .models import Post

#ListView mostra il contenuto sottoforma di lista


urlpatterns = [

    path('',ListView.as_view(

        queryset = Post.objects.all().order_by("-data"),
```

```
template_name = "listapost.html"),name="lista"),


    path('post-singolo',views.postSingolo,name="singolo"),

    path('contatti',views.contatti,name="contatti"),

]
```

15) Creare i templates (il template html descrive come i dati vengono presentati).

Creare la cartella templates nella cartella principale del progetto con all'interno i file html. Per aggiungere dei tempaltes nel database aprire file settings.py all'interno del progetto principale>>TEMPLATES>>DIRS>>e aggiungere [BASE_DIR / 'templates'].

La cartella templates conterrà listapost.html,contatti.html, post_singolo.html

16) Creare le views.py(view.py descrive quali dati dobbiamo vedere perché potrebbe esserci una sezione dedicata agli utenti iscritti che ovviamente devono accede e vedere dati maggiori)

queste views.py ricevono una richiesta e restituisce una risposta. La richiesta conterà l'url richiesto dall'utente e la risposta sarà la pagina Html che gli verrà mostrata.

```
from django.shortcuts import render

# Create your views here.

def listaPost(request):

    return render(request,"listapost.html")

def postSingolo(request):

    return render(request,"post singolo.html")

def contatti(request):

    return render(request,"contatti.html")
```

17) Modificare il template di listapost con il seguente codice:

```
<!DOCTYPE html>

<html>

<head><title>Lista Post</title></head>

<body>

        {% for object in object_list %}

<a href="/{{object.id}}/{{object.slug}}/" style="text-decoration: none;"><h1>{{object.titolo }}</h1></a><!--
object.id è la chiave primaria-->

<h3>{{ object.data }}</h3>

<p>{{ object.contenuto|truncatewords_html:50}}</p><!--per qualsiasi particolare di design crecare filtri-->

        {% endfor %}

</body>

</html>
```

18) Passiamo al post singolo utilizzando DetailView, per fare ciò entriamo nel file urls.py all'interno della nostra APP e modifichiamo la riga dei post singoli con seguente codice:

```
path('<int:id>/<slug:slug>/',DetailView.as_view(
    model = Post,
    template_name = "post singolo.html",),name="singolo"),
```

19) Modificare il template del file post singolo.html con il seguente codice:

```
<!DOCTYPE html>
<html>
<head><title>Post Singolo</title></head>
<body>

<h1>{{object.titolo }}</h1>
<h4>{{ object.data }}</h4>
<p>{{ object.contenuto}}</p>

</body>
</html>
```

20) Miglioriamo il collegamento con i post singoli con una best practice che serve ad irrobustire url per accedere ai post singoli utilizzando la funzione get_absolute_url() per prima cosa creiamo la funzione nei models.py all'interno della nostra APP inserendo il seguente codice:

```
from django.urls import reverse

# Best practice per ottenere l'url di un post singolo
def get_absolute_url(self):
    return reverse("singolo", kwargs={"id": self.id,"slug":self.slug})
```

21) Modificare la lista dei post sostituendo a /{{object.id}}/{{object.slug}}/ con {{object.get_absolute_url}}

22) Per migliorare l'esperienza di chi visita il sito e per evitare un sovraccarico di post utilizzeremo un sistema di paginazioni per far ciò bisognerà modificare nel file urls.py all'interno dell'APP aggiungendo alla path della lista post il seguente parametro:

```
paginate_by = 5
```

23) Aggiungere al template della listpost.html il seguente codice:

```
<div>
        {% if page_obj.has_previous %}
<a href="?page={{ page_obj.previous_page_number }}">Precedente</a>
        {% endif %}
```

```
        {% if page_obj.has_next %}
<a href="?page={{ page_obj.next_page_number }}">Successiva</a>
{% endif %}
</div>
```

24)per semplificare e alleggerire il programma si fa uso dell'ereditarietà dei templates che consiste nel creare un file come base.html che conterrà tutte quelle parti che possono essere risparmiate questo file va posizionato all'interno della cartella templates con il seguente codice:

```
<DOCTYPE html>
<html>
<head>
<title>{% block head_title %}My Django Blog{% endblock head_title %}</title>
</head>
<body>
<div class="container">
        {% block content %}
        {% endblock content %}
</div>
</body>
</html>
```

25) Adesso bisogna modificare gli altri template che saranno tutti figli di base.html

listapost.html avrà il seguente codice:

```
{% extends "base.html" %}
   {% block content %}
      {% for object in object_list %}
<a href="{{object.get_absolute_url}}" style="text-decoration: none;"><h1>{{object.titolo }}</h1></a><!--
object.id è la chiave primaria-->
<h3>{{ object.data }}</h3>
<p>{{object.contenuto|safe|linebreaks|truncatewords_html:50}}</p><!-Safe ci serve a interpretare codici Html
come tali invece linebreaks serve ad interpretare le righe vuote come tali -->
{% endfor %}
<div>
        {% if page_obj.has_previous %}
<a href="?page={{ page_obj.previous_page_number }}">Precedente</a>
        {% endif %}
        {% if page_obj.has_next %}
```

```
<a href="?page={{ page_obj.next_page_number }}">Successiva</a>
{% endif %}
</div>
   {% endblock content %}
```

post singolo.html avrà il seguente codice:

```
{% extends "base.html" %}
<title>{% block head_title %}{{ block.super }} | {{object.titolo }}{% endblock head_title %}</title>
   {% block content %}
<h1>{{object.titolo }}</h1>
<h4>{{ object.data }}</h4>
<p>{{ object.contenuto}}</p>
{% endblock content %}
```

contatti.html avrà il seguente codice:

```
{% extends "base.html" %}

<title>{% block head_title %}{{ block.super }} | Contatti{% endblock head_title %}</title>
{% block content %}
<h1>Contatti a questa Email</h1>
<h2>Email: mia mail</h2>
   {% endblock content %}
```

26) Andiamo ora ad aggiungere i file statici come immagini,CSS, JavaScript. Quando nei setting.py debug è settato su truestiamo dicendo a Django che stiamo ancora in fase di sviluppo e quindi i file statici vengono gestiti automaticamente grazie a 'django.contrib.staticfiles' che si trova nelle APP istallate per far si che questi file statici vengano identificati dobbiamo inserire alla fine del file settings.py il seguente codice:

```
STATICFILES_DIRS = [BASE_DIR / 'static']
```

27) creare nella cartella del progetto principale la cartella static con dentro le cartelle css,img,js dentro la cartella css creare un nuovo file chiamato mydjangoblog.css con il seguente codice:

```
h1{
   color: red;
}
```

28) Adesso caricare i file statici all'interno del template base.html aggiungendo dotto html:

```
{% load static %}
```

sotto title:

<link rel="stylesheet" href="{% static 'css/mydjangoblog.css' %}" />

29)bootstrap è un Framework utilizzato dal lato grafico e aiuta nel design,ci fornisce html CSS e javascript.

il CSS di bootstrap è basato sul Grid system ovvero il sistema a griglia responsive mobile quindi è stato pensato per essere adattato a qualsiasi schermo.

I componenti sono pezzi di codice html che bisogna solo copiare e incollare nel proprio codice.

JavaScript serve principalmente a dare vita ai componenti.

passiamo ora all'istallazione l'istallazione può avvenire in 2 modi diversi uno è scaricandolo tramite zip

l'altro modo invece è tramite CND(Content-Delivery-Network) che si tratta di un server che ci andrà a fornire il css è un sistema che permette al client che visita il sito di ottenere gli elementi statici in maniera più veloce

per fare ciò basta copiare e incollare dei link nei nostri template così poi quando un client visita il sito il server glieli scarica

link:

<!-- Latest compiled and minified CSS -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">

<!-- Optional theme -->

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp" crossorigin="anonymous">

<!--jquery-->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<!-- Latest compiled and minified JavaScript -->

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous"></script>

30) Passiamo ora a creare la navbar che grazie a bootstrap ne abbiamo gia una fatta per noi che va solo inserita nel base.html e modificata per farla apparire come si vuole

<nav class="navbar navbar-default">

<!-- Il marchio e l'interruttore vengono raggruppati per una migliore visualizzazione mobile -->

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-expanded="false">

Toggle navigation

</button>

Brand

<!-- Raccogli i collegamenti di navigazione, i moduli e altri contenuti per la commutazione -->

<ul class="nav navbar-nav">

<li class="active">Link (current)

Link

31) Passiamo alle modifiche aggiungendo un logo che viene poi visualizzato solo da desktop con il seguente codice:

<nav class="navbar navbar-default navbar-fixed-top">

<!-- Il marchio e l'interruttore vengono raggruppati per una migliore visualizzazione mobile -->

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-expanded="false">

Toggle navigation

</button>

My Django Blog

<!--hidden-xs-->

<!-- Raccogli i collegamenti di navigazione, i moduli e altri contenuti per la commutazione -->

<ul class="nav navbar-nav">

<li class="active">Home(current)

```
<li><a href="/contatti">Contatti</a></li>
</ul>
</div><!--navbar-collapse-->
</div><!--container-fluid-->
</nav><!--navbar-default-->
```

32) per alcuni problemi causati dall'aggiunta della classe navbar-fixed-top c'è bisogno di modificare un pò il CSS aggiungendo il seguente codice:

```
body
{
    padding-top: 70px;
}
```

33) Aggiungiamo un footer con il seguente codice:

```
<footer>
<div class="container-fluid">
<p style="color:white;  font-weight:bold">Canali Social:</p>
<a style="color:white" href="Link al Vostro Canale YouTube"><span class="glyphicon glyphicon-thumbs-up"></span>YouTube</a><br>
<a style="color:white" href="Link al Vostro Account Twitter"><span class="glyphicon glyphicon-thumbs-up"></span>Twitter</a><br>
<a style="color:white" href="Link al Vostro Profilo Facebook"><span class="glyphicon glyphicon-thumbs-up"></span>Facebook</a><br>
<br>
<a style="color:white" href="#">Torna Su</a><br>
</div>
</footer>
```

34) Diamogli un po' di stile con css aggiungendo il seguente codice:

```
.contenuto {
    min-height: 95%;
    padding-bottom: 40px;
}
```

(da aggiungere al div che contiene l'intera pagina)

```
footer {
    padding: 40px;
    background-color: #092E20;
text-align: center;
}
```

35) Quando aggiungiamo un’immagine nel sito ci sono 2 modi per renderla responsive

modo 1:

aggiungere alle immagini la classe: img-responsive

modo 2:

tramite css con il seguente codice:

```
img{
    max-width: 100%;
    height: auto;
display:block;
}
```

36) Andiamo ora ad organizzare la pagina con la Grid system responsive aagiungendo nel file base.html all'interno del contenuto della pagina il seguente codice:

```
<div class="row">

<div class="col-sm-9">

        {% block content %}

        {% endblock content %}

</div><!--col-sm-9-->

<div class="col-sm-2-offset-1" ><!--offset Questa classe aumenta il margine sinistro di una colonna spostandola più a destra-->
<center>
<h4>My Django Blog</h4>
<h5>Creato da Giuseppe Saccavino in collaborazione con Alessandro Esposito</h5>
<p>Powered by <strong><i>Django & Bootstrap</i></strong></p>
</center>
</div><!--col-sm-2-offset-1-->

</div><!--row-->
```

37) Andiamo a migliorare i bottoni della paginazione sostituendo il codice vecchio che si trova nella lista post.html con il seguente codice:

```
<div>
<nav aria-label="...">
<ul class="pager pull-right">
        {% if page_obj.has_previous %}
<li><a href="?page={{ page_obj.previous_page_number }}">Precedente</a></li>
        {% endif %}

        {% if page_obj.has_next %}
<li><a href="?page={{ page_obj.next_page_number }}">Successiva</a></li>
        {% endif %}
</ul>
</nav>
</div>
```

38) Per risolvere il problema della sovrapposizione per via del sistema responsive basta aggiungere nel div che contiene i bottoni della paginazione e aggiungere la seguente classe:

clearfix

39) Ora miglioriamo un altro aspetto grafico che riguarda la navbar aggiungendo un comportamento (tramite python e non javascript) tramite le seguenti classi appartenenti al bottone home e contatti:

```
class="{% if request.path_info == '/' %} active {% endif %}"
class="{% if request.path_info == '/contatti' %} active {% endif %}"
```

40) Passiamo un po' a l'ultima parte del progetto vedendo finalmente come pubblicare online il nostro sito, per mandare online il nostro sito abbiamo bisogno di alcuni strumenti:

1. client SSH che ci servirà per connetterci tramite riga di commando al server su cui poi andremo a caricare il nostro sito ovvero il server che andrà ad ospitare il nostro sito (questa operazione si può evitare su Linux e OS perché lo hanno già integrato). Da Windows si può usare Putty

2. software Per quanto riguarda il trasferimento dei file, quindi il passaggio in cui copieremo i file del nostro blog per poi caricarli sul server per fare ciò useremo un programma ad Interfaccia Grafica chiamato FileZilla

questi strumenti serviranno poi per pubblicarlo su DIGITAL OCEAN perché a differenza di altri usa un servizio chiamato "one click in store" che ha a disposizione delle macchine preconfezionata per quanto riguarda Django.

Una volta iscritti quindi selezioniamo una macchina preconfezionata, scegliamo lo spazio quello da 5$ al mese, poi bisognerebbe scegliere il datacenter quindi la locazione (Amsterdam),si potrebbe sceglie di fare i backup che sono automatici gestiti da DIGITAL OCEAN.

Una volta compilati questi campi ci verrà inviata l'email con le credenziali per l'accesso e l'indirizzo IP della macchina che abbiamo creato. Questo indirizzo IP porterà alla schermata iniziale di creazione la stessa che abbiamo visto al primo avvio del progetto. Infine con Putty e FileZilla avremmo caricato il sito online.