ENG 346
Data Structures and Algorithms for Artificial Intelligence
Python Basics

Dr. Mehmet PEKMEZCİ

mpekmezci@gtu.edu.tr

https://github.com/mehmetpekmezci/GTU-ENG-346

Rules

- Write READABLE and EASY-TO-MAINTAIN code !!!
  - **Self Descriptive Code :** Always use long descriptive names for functions/classes/variables. Do not use i,j ….
  - All variables are on top (Class/function or python file)
  - All imports are on top.
  - Do not use numbers directly, first assign them to a variable.

    (e.g. ~~if distance>100,~~ MAX_DISTANCE=100;if distance >MAX_DISTANCE)
  - Constant variable names are all capital (e.g. MAX_DISTANCE)
- Minimum usage of global variables, always use local variables.

About Python

- Python is an *interpreted*, *object-oriented* and *high-level* programming language.
  - C/C++/Rust/GO → compiled into binary (machine language)
  - Java → compiled into byte code, then byte codes are interpreted by jvm
  - Perl/PHP/Python/Shell Scripts (Bash) → Directly interpreted by their interpreter programs.
- Created by Guido van Rossum, a Dutch programmer, in 1989.
- First public release: Python 0.9.0 in 1991.
- Python 2.0 released in 2000.
- Python 3.0 released in 2008.

How to Access Python Interpreter?

- Official Website: https://www.python.org/
  - Interpreter + pip
  - Jupyter notebook via “jupyter” package
- Anaconda: https://anaconda.org/
  - Interpreter + conda + pip
  - Jupyter notebook via “jupyter” package
- Google Colab: https://colab.research.google.com/
  - Via Colab notebooks (like Jupyter notebooks)
- Kaggle: https://www.kaggle.com/
  - Via Kaggle notebooks (like Jupyter notbooks)

OR - Windows Subsystem for Linux (WSL)

```
## https://learn.microsoft.com/en-us/windows/wsl/install
## https://ubuntu.com/desktop/wsl

## In poweshell :

Wsl --list --online

Wsl --install Ubuntu-24.04

Wsl -d Ubuntu-24.04


## Linux Shell Starts Here

sudo apt update
sudo apt install python3-pip
pip3 install matplotlib numpy pandas

## The first time you run code from Ubuntu, it will trigger a download of the necessary
dependencies:
code .
```

Python Basics

- Variable naming: case-sensitive (e.g., temp vs Temp vs TEMP)
- Reserved words: cannot be used as variables.
  - False as continue else from in not return yield None assert def except global is or try True break del finally if lambda pass while and class elif for import nonlocal raise with
- Dynamically typed: No declarations are needed for the variables.
- Everything is basically an Object.
- Indentation is very important !

Indentation

CORRECT

```
import sys

def main():
   if len(sys.argv) < 2 :
       print("Usage: python3 main.py
<first_argument>")
       exit(1)
   FIRST_ARGUMENT=float(sys.argv[1])
   COEFFICIENT=6.75
   RESULT=COEFFICIENT*FIRST_ARGUMENT
   print(f"RESULT={RESULT}")
   USER_INPUT=input("Enter Another VALUE: ")
   RESULT=COEFFICIENT*float(USER_INPUT)
   print(f"2. RESULT={RESULT}")

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

INCORRECT

```
import sys

def main():
   if len(sys.argv) < 2 :
       print("Usage: python3 main.py
<first_argument>")
       exit(1)
   FIRST_ARGUMENT=float(sys.argv[1])
   COEFFICIENT=6.75
   RESULT=COEFFICIENT*FIRST_ARGUMENT
   print(f"RESULT={RESULT}")
 USER_INPUT=input("Enter Another VALUE: ")
   RESULT=COEFFICIENT*float(USER_INPUT)
   print(f"2. RESULT={RESULT}")

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

Arguments and User Input

main.py

```python
import sys

def main():
    if len(sys.argv) < 2 :
        print("Usage: python3 main.py <first_argument>")
        exit(1)
    FIRST_ARGUMENT=float(sys.argv[1])
    COEFFICIENT=6.75
    RESULT=COEFFICIENT*FIRST_ARGUMENT
    print(f"RESULT={RESULT}")
    USER_INPUT=input("Enter Another VALUE: ")
    RESULT=COEFFICIENT*float(USER_INPUT)
    print(f"2. RESULT={RESULT}")

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

OUTPUT

```
mpekmezci@cobalt:/tmp$ python3 main.py
Usage: python3 main.py <first_argument>
mpekmezci@cobalt:/tmp$ python3 main.py 7
RESULT=47.25
Enter Another VALUE: 8
2. RESULT=54.0
mpekmezci@cobalt:/tmp$
```

Lists and Dictionaries

GEBZE TECHNICAL UNIVERSITY

main.py

```python
def main():
    LIST1=[ 1,3,8,4 ]
    print(f"LIST1:{LIST1}")
    print(f"len(LIST1):{len(LIST1)}")
    LIST1.append("abc")
    print(f"LIST1:{LIST1}")
    print(f"LIST1[4]:{LIST1[4]}\n")

    DICT1={
            "key1":"value1",
            "key2":{
               "key2.1":"value2.1",
               "key2.2":"value2.2"
               }
            }
    print(f"DICT1:{DICT1}")
    print(f"DICT1.keys():{DICT1.keys()}")
    print(f"DICT1[key1]:{DICT1['key1']}")
    DICT1["key3"]="value3"
    print(f"DICT1:{DICT1}")

if __name__ == '__main__':
 main()

```

OUTPUT

```
mpekmezci@cobalt:/tmp$ python3 main.py
LIST1:[1, 3, 8, 4]
len(LIST1):4
LIST1:[1, 3, 8, 4, 'abc']
LIST1[4]:abc

DICT1:{'key1': 'value1', 'key2': {'key2.1': 'value2.1', 'key2.2': 'value2.2'}}
DICT1.keys():dict_keys(['key1', 'key2'])
DICT1[key1]:value1
DICT1:{'key1': 'value1', 'key2': {'key2.1': 'value2.1', 'key2.2': 'value2.2'}, 'key3': 'value3'}
```

For loop

GEBZE TECHNICAL UNIVERSITY

main.py

```python
import time
import datetime

def main():

    LIST1=[ 1,3,8,"deneme" ]

    for value in LIST1 : print(f"{value}")

    for index in range(len(LIST1)) :
        print(f"LIST1[{index}]={LIST1[index]}")

    DICT1={
            "key1":"value1",
            "key2":{"key2.1":"value2.1","key2.2":"value2.2"}
           }

    for key in DICT1.keys() :
        print(f"DICT1[{key}]={DICT1[key]}")


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

OUTPUT

```
mpekmezci@cobalt:/tmp$ python3 main.py
1
3
8
deneme
LIST1[0]=1
LIST1[1]=3
LIST1[2]=8
LIST1[3]=deneme
DICT1[key1]=value1
DICT1[key2]={'key2.1': 'value2.1', 'key2.2': 'value2.2'}
mpekmezci@cobalt:/tmp$
```

While loop, timestamp, datetime, sleep

GEBZE TECHNICAL UNIVERSITY

main.py

```python
import time
import datetime

def main():
    LIST1=[ 1,3,8,"deneme" ]

    i=0
    while i<5 :
        print(i)
        print(
            "Current Time (human Readable) :"
            +str(datetime.datetime.now())
            )
        print(
            "Current timestamp:"
            +str(time.time())
            )
        i=i+1
        time.sleep(1)


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

OUTPUT

```
mpekmezci@cobalt:/tmp$ python3 main.py
0
Current Time (human Readable) :2025-08-23 18:49:36.741377
Current timestamp:1755964176.7414153
1
Current Time (human Readable) :2025-08-23 18:49:37.741626
Current timestamp:1755964177.741657
2
Current Time (human Readable) :2025-08-23 18:49:38.741893
Current timestamp:1755964178.7419362
3
Current Time (human Readable) :2025-08-23 18:49:39.742067
Current timestamp:1755964179.742092
4
Current Time (human Readable) :2025-08-23 18:49:40.742298
Current timestamp:1755964180.7423482
mpekmezci@cobalt:/tmp$
```

String operations

main.py

```python
def main():
    my_string = 'We are roots of havelsan :)'
    company = my_string[16:24]
    print(company)
    print(my_string.split())
    if "root" in my_string:
        print("This string contains root word")
    if my_string.startswith("We"):
        print("This string starts with 'We'")
    if my_string.endswith(":)"):
        print("This string ends with ':)'")

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

OUTPUT

```
mpekmezci@cobalt:/tmp$ python3 main.py
havelsan
['We', 'are', 'roots', 'of', 'havelsan', ':)']
This string contains root word
This string starts with 'We'
This string ends with ':)'
mpekmezci@cobalt:/tmp$
```

File operations

main.py

```python
def main():
    with open("deneme.txt", "r") as f:
        data = f.read().splitlines()
    print(data)
    print("--------------------------")
    for line in data:
        print(line)
    print("--------------------------")
    record_file = open("./deneme.txt", "a")
    record_file.write("We are roots of Havelsan 1\n")
    record_file.write("We are roots of Havelsan 2\n")
    record_file.close()
    record_file = open("./deneme.txt", "r")
    print(record_file.read())
    record_file.close()

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

OUTPUT

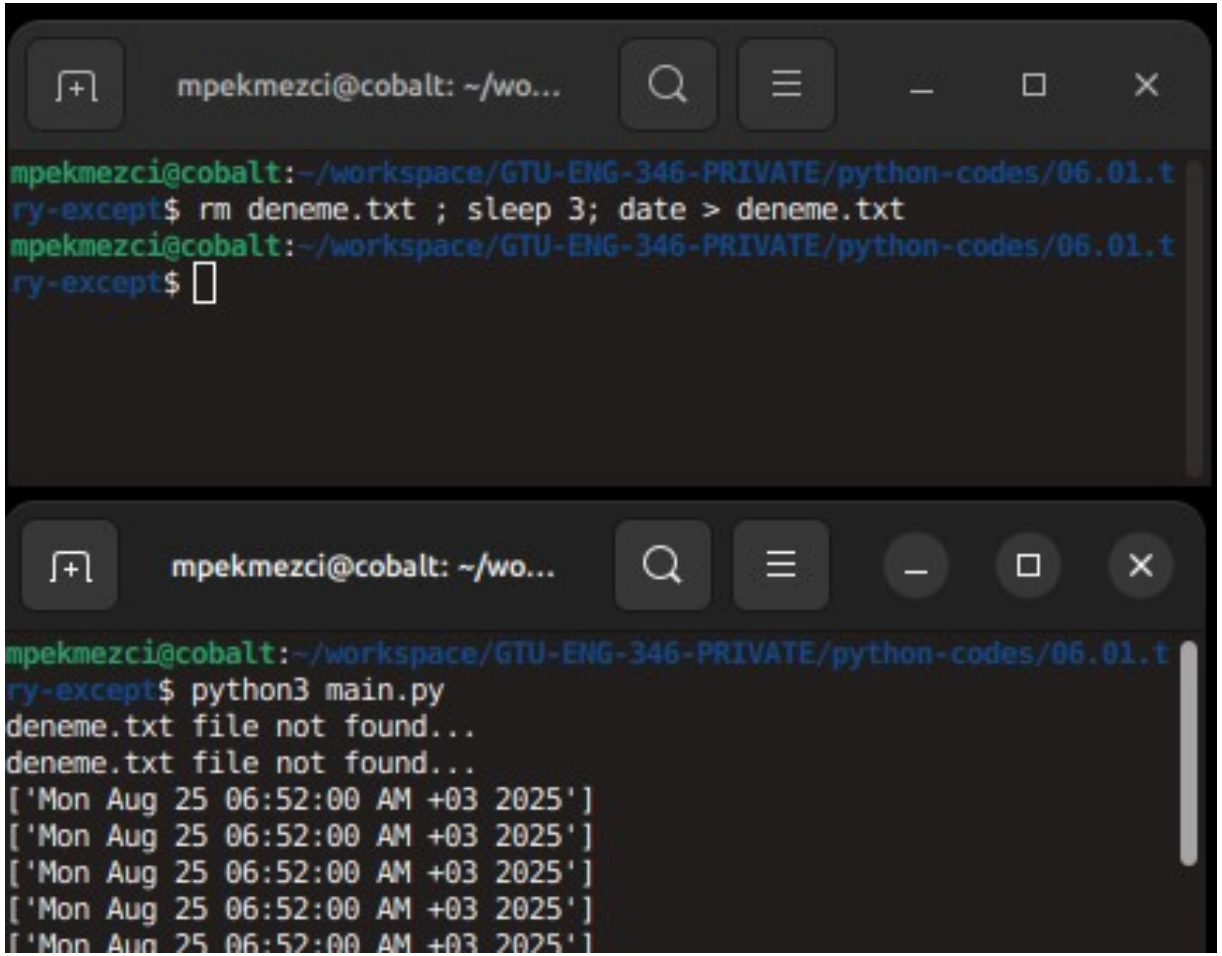
```
mpekmezci@cobalt:/tmp$ echo "deneme 1 line" > deneme.txt
mpekmezci@cobalt:/tmp$ python3 main.py
['deneme 1 line']
--------------------------
deneme 1 line
--------------------------
deneme 1 line
We are roots of Havelsan 1
We are roots of Havelsan 2

mpekmezci@cobalt:/tmp$
```

Try-Except

main.py

```python
import time

def main():
  while True:
    try:
      with open("deneme.txt", "r") as f:
        data = f.read().splitlines()
      print(data)
      if "exit" in data:
          exit(0)
    except FileNotFoundError:
      print("deneme.txt file not found...")
    time.sleep(1)

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

OUTPUT



Object Oriented Programming

GEBZE TECHNICAL UNIVERSITY

main.py

```python
#!/usr/bin/python3

from GWNode  import GWNode
from Node  import Node

def main():
  nodeList=[]

  try:
   with open("hosts", "r") as f:
       data = f.read().splitlines()
  except FileNotFoundError :
       print("Can not find a ./hosts file containing "+
             "'ip,hostname' list of nodes")
       exit(1)
  for line in data:
      ip_and_name=line.split(',')
      ip=ip_and_name[0]
      name=ip_and_name[1]
      if ip.startswith("127"):
         nodeList.append(GWNode(ip,name,"gateway"))
      else:
         nodeList.append(Node(ip,name))
  for node in nodeList:
      print(f"Checking if we can reach {node.name}" +
            f"by {node.ip} at port 22 , this is a"+
            f"node of type :{node.getNodeType()}"
            )
      if node.checkSSH() :
          print(f"Yes we can reach {node.name}"+
                f"by {node.ip} at port 22")
      else:
          print(f"Sorry we can not reach {node.name} by"+
                f"{node.ip} at port 22")


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

Node.py

```python
import socket

class Node :
    def __init__(self, ip, nodeName):
        self.name=nodeName
        self.ip=ip
        self.sshport=22
        self.timeout=5 # seconds

    def checkSSH(self, alternative_port_non_mandatory_argument=None):
      try:
        socket.setdefaulttimeout(self.timeout)
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        if alternative_port_non_mandatory_argument is not None:

          s.connect(
            (self.ip,alternative_port_non_mandatory_argument)
           )
        else:
          s.connect((self.ip,self.sshport))

      except OSError as error:
        return False
      else:
        s.close()
        return True

    def getNodeType(self):
         return "Standard Node"
```

Object Oriented Programming

GEBZE TECHNICAL UNIVERSITY

GWNode.py

```python
from Node import Node

class GWNode(Node) :
    def __init__(self, ip, nodeName,gw_property):
        Node.__init__(self,  ip, nodeName)
        self.gw_property=gw_property

    def getNodeType(self):
         return "Gateway Node"
```

hosts

```
127.0.0.1,gateway_01
10.1.10.101,another_computer
```

OUTPUT

```
mpekmezci@cobalt:~/workspace/GTU-ENG-346-PRIVATE/python-codes/06.oop$ python3 main.py
Checking if we can reach gateway_01 by 127.0.0.1 at port 22 , this is a node of type :Gateway Node
Yes we can reach gateway_01 by 127.0.0.1 at port 22
Checking if we can reach another_computer by 10.1.10.101 at port 22 , this is a node of type :Standard Node
Sorry we can not reach another_computer by 10.1.10.101 at port 22
mpekmezci@cobalt:~/workspace/GTU-ENG-346-PRIVATE/python-codes/06.oop$
```

Numpy

GEBZE TECHNICAL UNIVERSITY

main.py

```python
import numpy as np
a = np.array([1, 2, 3])
print(f"type(a)={type(a)}")
print(a.shape)
print(a[0], a[1], a[2])
a[0] = 5
print(f"a={a}")
b = np.array([[1,2,3],[4,5,6]])
print(f"b.shape={b.shape}")
print(f"b[0, 0]={b[0, 0]}, b[0, 2]={b[0,2]}")
print(f"np.matmul(b.T,b)={np.matmul(b.T,b)}")
print(f"np.zeros((2,2))={np.zeros((2,2))}")
print(f"np.ones((1,2))={np.ones((1,2))}")
print(f"np.full((2,2), 7)={np.full((2,2), 7)}")
d = np.eye(2)
print(f"d={d}")
e = np.random.random((2,2))
print(f"e={e}")
print(f"d*e:{d*e}")
print(f"np.matmul(d,e):{np.matmul(d,e)}")
f=np.concatenate((d, e), axis=0)
print(f"f={f}")
g=np.concatenate((d, e), axis=1)
print(f"g={g}")
data = np.array([1.0, 2.0])
data = data * 1.6
print(f"data={data}")
```

OUTPUT

```
mpekmezci@cobalt:~/workspace/GTU-ENG-346-PRIVATE/python-
type(a)=<class 'numpy.ndarray'>
(3,)
1 2 3
a=[5 2 3]
b.shape=(2, 3)
b[0, 0]=1, b[0, 2]=3
np.matmul(b.T,b)=[[17 22 27]
 [22 29 36]
 [27 36 45]]
np.zeros((2,2))=[[0. 0.]
 [0. 0.]]
np.ones((1,2))=[[1. 1.]]
np.full((2,2), 7)=[[7 7]
 [7 7]]
d=[[1. 0.]
 [0. 1.]]
e=[[0.75065747 0.92162586]
 [0.77748988 0.52421712]]
d*e:[[0.75065747 0.        ]
 [0.         0.52421712]]
np.matmul(d,e):[[0.75065747 0.92162586]
 [0.77748988 0.52421712]]
f=[[1.         0.        ]
 [0.         1.        ]
 [0.75065747 0.92162586]
 [0.77748988 0.52421712]]
g=[[1.         0.         0.75065747 0.92162586]
 [0.         1.         0.77748988 0.52421712]]
data=[1.6 3.2]
mpekmezci@cobalt:~/workspace/GTU-ENG-346-PRIVATE/python-
```

CSV

main.py

```python
import csv

with open('employee_birthday.txt') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            print(f'Column names are {", ".join(row)}')
            line_count += 1
        else:
            print(f'\t{row[0]} works in the {row[1]} department, and was born in {row[2]}.')
            line_count += 1
    print(f'Processed {line_count} lines.')


with open('employee_file.csv', mode='w') as employee_file:
    employee_writer = csv.writer(employee_file, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)

    employee_writer.writerow(['John Smith', 'Accounting', 'November'])
    employee_writer.writerow(['Erica Meyers', 'IT', 'March'])
```

employee_birthday.txt

```
mpekmezci@cobalt:~/workspace/GTU-ENG-346-PRIVATE/python-codes/07.02-csv$ cat employee_birthday.txt
name,department,birthday month
John Smith,Accounting,November
Erica Meyers,IT,March
mpekmezci@cobalt:~/workspace/GTU-ENG-346-PRIVATE/python-codes/07.02-csv$
```

employee_file.csv

	A	B	C
1	John Smith	Accounting	November
2	Erica Meyers	IT	March
3			
4			
5			
6			

Pickle

main.py

```python
import pickle

a = {'hello': 'world'}

with open('filename.pickle', 'wb') as handle:
    pickle.dump(a, handle, protocol=pickle.HIGHEST_PROTOCOL)

with open('filename.pickle', 'rb') as handle:
    b = pickle.load(handle)

print(a == b)
```

OUTPUT

```
mpekmezci@cobalt:~/workspace/GTU-ENG-346-PRIVATE/python-codes/07.03-pickle$ python3 main.py
True
mpekmezci@cobalt:~/workspace/GTU-ENG-346-PRIVATE/python-codes/07.03-pickle$
```

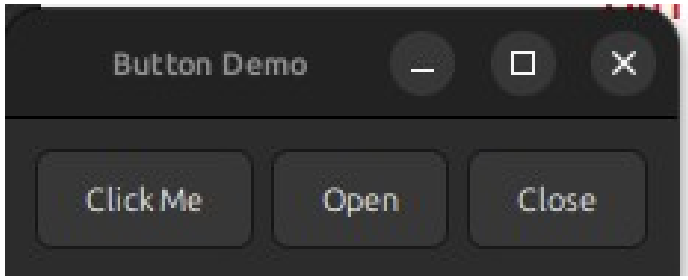
Python GUI - GTK - Button

GEBZE TECHNICAL UNIVERSITY

main.py

```python
import gi

gi.require_version("Gtk", "3.0")
from gi.repository import Gtk


class ButtonWindow(Gtk.Window):
    def __init__(self):
        super().__init__(title="Button Demo")
        self.set_border_width(10)

        hbox = Gtk.Box(spacing=6)
        self.add(hbox)

        button = Gtk.Button.new_with_label("Click Me")
        button.connect("clicked", self.on_click_me_clicked)
        hbox.pack_start(button, True, True, 0)

        button = Gtk.Button.new_with_mnemonic("_Open")
        button.connect("clicked", self.on_open_clicked)
        hbox.pack_start(button, True, True, 0)

        button = Gtk.Button.new_with_mnemonic("_Close")
        button.connect("clicked", self.on_close_clicked)
        hbox.pack_start(button, True, True, 0)

    def on_click_me_clicked(self, button):
        print('"Click me" button was clicked')

    def on_open_clicked(self, button):
        print('"Open" button was clicked')

    def on_close_clicked(self, button):
        print("Closing application")
        Gtk.main_quit()


win = ButtonWindow()
win.connect("destroy", Gtk.main_quit)
win.show_all()
Gtk.main()
```

OUTPUT



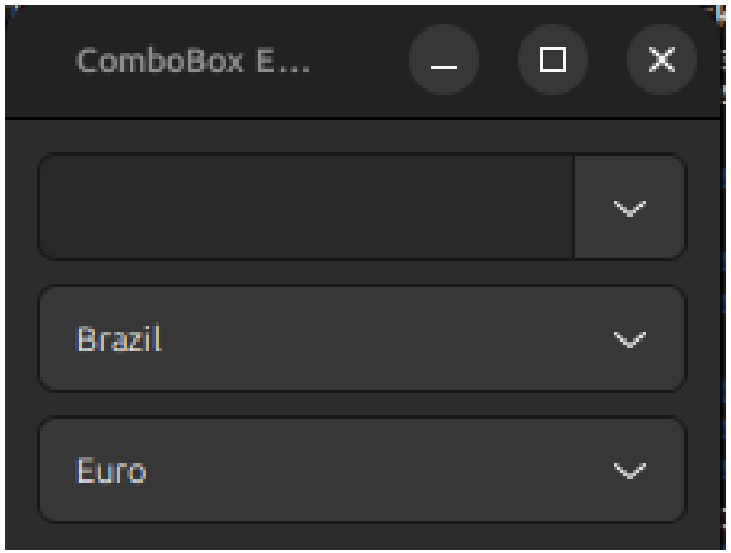

```
mpekmezci@cobalt:~/workspace/GTU-ENG-346-PRIVATE/python-codes/15.gnome$ python3 button.py
"Click me" button was clicked
"Open" button was clicked
Closing application
```

Python GUI - GTK - Combobox

main.py

```python
import gi
gi.require_version("Gtk", "3.0")
from gi.repository import Gtk
class ComboBoxWindow(Gtk.Window):
    def __init__(self):
        super().__init__(title="ComboBox Example")
        self.set_border_width(10)
        name_store = Gtk.ListStore(int, str)
        name_store.append([1, "Billy Bob"])
        name_store.append([11, "Billy Bob Junior"])
        name_store.append([31, "Xavier McRoberts"])
        vbox = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=6)
        name_combo = Gtk.ComboBox.new_with_model_and_entry(name_store)
        name_combo.connect("changed", self.on_name_combo_changed)
        name_combo.set_entry_text_column(1)
        vbox.pack_start(name_combo, False, False, 0)
        country_store = Gtk.ListStore(str)
        countries = ["Austria","Brazil","Uruguay",]
        for country in countries:
            country_store.append([country])
        country_combo = Gtk.ComboBox.new_with_model(country_store)
        country_combo.connect("changed", self.on_country_combo_changed)
        renderer_text = Gtk.CellRendererText()
        country_combo.pack_start(renderer_text, True)
        country_combo.add_attribute(renderer_text, "text", 0)
        vbox.pack_start(country_combo, False, False, True)
        currencies = ["Euro","US Dollars","Swiss franc",]
        currency_combo = Gtk.ComboBoxText()
        currency_combo.set_entry_text_column(0)
        currency_combo.connect("changed", self.on_currency_combo_changed)
        for currency in currencies:
            currency_combo.append_text(currency)
        currency_combo.set_active(0)
        vbox.pack_start(currency_combo, False, False, 0)
        self.add(vbox)

    def on_name_combo_changed(self, combo):
        tree_iter = combo.get_active_iter()
        if tree_iter is not None:
```

OUTPUT

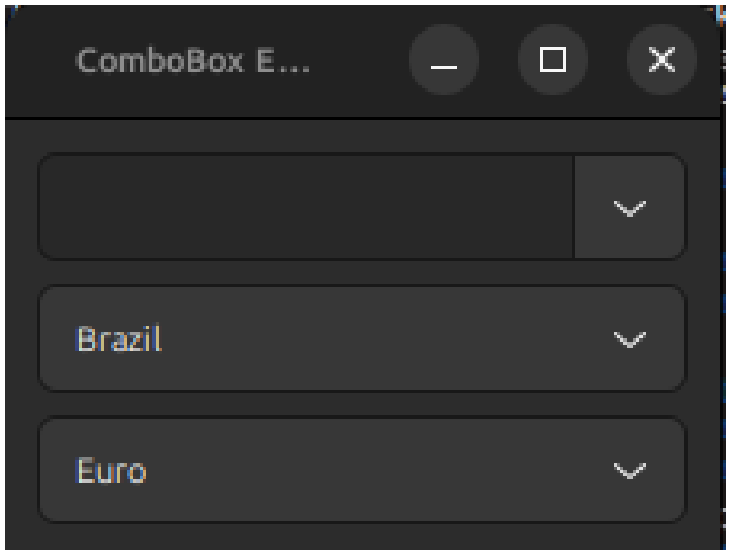


Python GUI - GTK - Combobox

main.py

```
    def on_name_combo_changed(self, combo):
        tree_iter = combo.get_active_iter()
        if tree_iter is not None:
            model = combo.get_model()
            row_id, name = model[tree_iter][:2]
            print("Selected: ID=%d, name=%s" % (row_id, name))
        else:
            entry = combo.get_child()
            print("Entered: %s" % entry.get_text())

    def on_country_combo_changed(self, combo):
        tree_iter = combo.get_active_iter()
        if tree_iter is not None:
            model = combo.get_model()
            country = model[tree_iter][0]
            print("Selected: country=%s" % country)

    def on_currency_combo_changed(self, combo):
        text = combo.get_active_text()
        if text is not None:
            print("Selected: currency=%s" % text)


win = ComboBoxWindow()
win.connect("destroy", Gtk.main_quit)
win.show_all()
Gtk.main()
```

OUTPUT



Python GUI - GTK - Grid

main.py

```python
import gi

gi.require_version("Gtk", "3.0")
from gi.repository import Gtk


class GridWindow(Gtk.Window):
    def __init__(self):

        super().__init__(title="Grid Example")

        button1 = Gtk.Button(label="Button 1")
        button2 = Gtk.Button(label="Button 2")
        button3 = Gtk.Button(label="Button 3")
        button4 = Gtk.Button(label="Button 4")
        button5 = Gtk.Button(label="Button 5")
        button6 = Gtk.Button(label="Button 6")

        grid = Gtk.Grid()
        grid.add(button1)
        grid.attach(button2, 1, 0, 2, 1)
        grid.attach_next_to(button3, button1, Gtk.PositionType.BOTTOM, 1, 2)
        grid.attach_next_to(button4, button3, Gtk.PositionType.RIGHT, 2, 1)
        grid.attach(button5, 1, 2, 1, 1)
        grid.attach_next_to(button6, button5, Gtk.PositionType.RIGHT, 1, 1)

        self.add(grid)


win = GridWindow()
win.connect("destroy", Gtk.main_quit)
win.show_all()
Gtk.main()
```

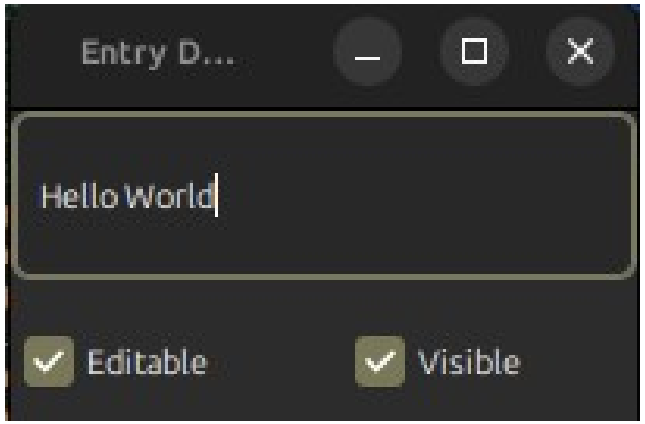
OUTPUT

Python GUI - GTK – Input Text

main.py

```python
import gi

gi.require_version("Gtk", "3.0")
from gi.repository import Gtk, GLib


class EntryWindow(Gtk.Window):
    def __init__(self):
        super().__init__(title="Entry Demo")
        self.set_size_request(200, 100)

        self.timeout_id = None

        vbox = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=6)
        self.add(vbox)

        self.entry = Gtk.Entry()
        self.entry.set_text("Hello World")
        vbox.pack_start(self.entry, True, True, 0)

        hbox = Gtk.Box(spacing=6)
        vbox.pack_start(hbox, True, True, 0)

        self.check_editable = Gtk.CheckButton(label="Editable")
        self.check_editable.connect("toggled", self.on_editable_toggled)
        self.check_editable.set_active(True)
        hbox.pack_start(self.check_editable, True, True, 0)

        self.check_visible = Gtk.CheckButton(label="Visible")
        self.check_visible.connect("toggled", self.on_visible_toggled)
        self.check_visible.set_active(True)
        hbox.pack_start(self.check_visible, True, True, 0)


    def on_editable_toggled(self, button):
        value = button.get_active()
        self.entry.set_editable(value)

    def on_visible_toggled(self, button):
        value = button.get_active()
        self.entry.set_visibility(value)



win = EntryWindow()
win.connect("destroy", Gtk.main_quit)
win.show_all()
Gtk.main()
```

OUTPUT



Built-in Types

Type	Description
bool	Boolean value
int	Integer
float	Floating-point number
list	Mutable sequence of objects
tuple	Immutable sequence of objects
str	Character string
set	Unordered set of distinct objects
frozenset	Immutable form of set class
dict	Dictionary (Associative mapping)

Operators

Operator Precedence		
	Type	**Symbols**
1	member access	expr.member
2	function/method calls container subscripts/slices	expr(...) expr[...]
3	exponentiation	**
4	unary operators	+expr, −expr, ~expr
5	multiplication, division	*, /, //, %
6	addition, subtraction	+, −
7	bitwise shifting	<<, >>
8	bitwise-and	&
9	bitwise-xor	^
10	bitwise-or	|
11	comparisons containment	**is**, **is not**, ==, !=, <, <=, >, >= **in**, **not in**
12	logical-not	**not** expr
13	logical-and	**and**
14	logical-or	**or**
15	conditional	val1 **if** cond **else** val2
16	assignments	=, +=, −=, *=, etc.

Control Flow

```
if first condition:
    first body
elif second condition:
    second body
elif third condition:
    third body
else:
    fourth body
```

Loops

```
while condition:
    body
```

```
for element in iterable:
    body
```

Functions

```
def count(data, target):
    n=0
    for item in data:
        if item == target: # found a
match
            n += 1

    return n
```

Built-in Funct

Common Built-In Functions	
Calling Syntax	**Description**
abs(x)	Return the absolute value of a number.
all(iterable)	Return True if bool(e) is True for each element e.
any(iterable)	Return True if bool(e) is True for at least one element e.
chr(integer)	Return a one-character string with the given Unicode code point.
divmod(x, y)	Return (x // y, x % y) as tuple, if x and y are integers.
hash(obj)	Return an integer hash value for the object (see Chapter 10).
id(obj)	Return the unique integer serving as an "identity" for the object.
input(prompt)	Return a string from standard input; the prompt is optional.
isinstance(obj, cls)	Determine if obj is an instance of the class (or a subclass).
iter(iterable)	Return a new iterator object for the parameter (see Section 1.8).
len(iterable)	Return the number of elements in the given iteration.
map(f, iter1, iter2, ...)	Return an iterator yielding the result of function calls f(e1, e2, ...) for respective elements e1 ∈ iter1, e2 ∈ iter2, ...
max(iterable)	Return the largest element of the given iteration.
max(a, b, c, ...)	Return the largest of the arguments.
min(iterable)	Return the smallest element of the given iteration.
min(a, b, c, ...)	Return the smallest of the arguments.
next(iterator)	Return the next element reported by the iterator (see Section 1.8).
open(filename, mode)	Open a file with the given name and access mode.
ord(char)	Return the Unicode code point of the given character.
pow(x, y)	Return the value x^y (as an integer if x and y are integers); equivalent to x ** y.
pow(x, y, z)	Return the value $(x^y \bmod z)$ as an integer.
print(obj1, obj2, ...)	Print the arguments, with separating spaces and trailing newline.
range(stop)	Construct an iteration of values $0, 1, \ldots,$ stop $- 1$.
range(start, stop)	Construct an iteration of values start, start $+ 1, \ldots,$ stop $- 1$.