```python
# Variables

x = 10
y = "Pakistan"
z = 3.3

#Data types

my_str = "Peshawar"
my_float = 96.9
my_integer = 154
my_bool = True

# type casting

a = int (my_float) # converting float to Integer
print(a)
# print type of variable
print(type(a))

b = str (my_float) # converting float to string
print(type(a))


num_1 = 10
num_2 = 20

# Arthimatic Operators
# Arithmetic operators are used with numeric values to perform common
mathematical operations:
# "+" , "-" , "*" , "/"

total = num_1 + num_2
print(total)
total = num_1 * num_2
print(total)
total = num_2 / num_1
print(total)
print(int(total))
total = num_1 - num_2
print(total)

# Assignment operators
# Assignment operators are used to assign values to variables:
# *= , -= , = , *=
```

```
num_1 += num_2 # is same as num_1 = num_1 + num_2
num_1 -= num_2 # is same as num_1 = num_1 - num_2
num_1 *= num_2 # is same as num_1 = num_1 * num_2

# Python Comparison Operators
# Comparison operators are used to compare two values:
# == , > , < , >= <=

result = (num_2 == num_2)
print(result)
result = (num_2 < num_1)
print(result)
result = (num_2 > num_1)
print(result) # will return bool False

# Lists
# Lists are used to store multiple items in a single variable.
# List is a collection which is ordered and changeable. Allows duplicate members.
# Lists are created using square brackets:

thislist = ["apple", "banana", "cherry", "banana"]
list1 = ["abc", 34, True, 40, "male"]
print(thislist)
print(thislist[1]) # will output banana
print(thislist[-2]) # will output cherry


thislist[1] = "Pakistan"
print(thislist[1]) # will output Pakistan
print(thislist) # will output list

# Tuple
# Tuple is a collection which is ordered and unchangeable. Allows duplicate
members.
# Tuples are written with round brackets.

mytuple = ("apple", "banana", "cherry")
tuple1 = ("abc", 34, True, 40, "male")

print(tuple1)
print(mytuple)
print(mytuple[2])
```

```python
# Set
# Sets are used to store multiple items in a single variable.
# Sets are written with curly brackets.
# Sets cannot have two items with the same value.

thisset = {"apple", "banana", "cherry", True, 1, 2}
# True and 1 is considered the same value:
print(thisset)
# Set is a collection which is unordered, unchangeable*, and unindexed. No
duplicate members.

# Dictionary
# Dictionaries are used to store data values in key:value pairs.
# A dictionary is a collection which is ordered*, changeable and do not allow
duplicates.


thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

print(thisdict["brand"])


# Python Collections (Arrays)
# There are four collection data types in the Python programming language:

# List is a collection which is ordered and changeable. Allows duplicate members.
# Tuple is a collection which is ordered and unchangeable. Allows duplicate
members.
# Set is a collection which is unordered, unchangeable*, and unindexed. No
duplicate members.
# Dictionary is a collection which is ordered** and changeable. No duplicate
members.


# Python If ... Else

a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
```

```
  print("a and b are equal")
else:
  print("a is greater than b")


# Python While Loops

# Print i as long as i is less than 6:

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

#   Python For Loops
# A for loop is used for iterating over a sequence (that is either a list, a
tuple, a dictionary, a set, or a string).

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

# Looping Through a String
for x in "banana":
  print(x)


# The range() function returns a sequence of numbers, starting from 0 by default,
and increments by 1 (by default), and ends at a specified number.

# Example
# Using the range() function:

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

# Python Functions
# A function is a block of code which only runs when it is called.

# You can pass data, known as parameters, into a function.

# A function can return data as a result.

def my_function():
  print("Hello from a function")
```

```python
# calling function

my_function()


# Python Lambda
# A lambda function is a small anonymous function.

# A lambda function can take any number of arguments, but can only have one
expression.
# Syntax
# lambda arguments : expression

x = lambda a, b : a * b
print(x(5, 6))
x = lambda a, b, c : a + b + c
print(x(5, 6, 2))

# o make a function that always doubles the number you send in:

# Example
def myfunc(n):
  return lambda a : a * n

mydoubler = myfunc(2)

print(mydoubler(11))

#creating a connection to the database.

import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
  port = "3306",
  database = "test"
)

print("Connection successful to Database :")
```

```
# Creating a Table

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (name VARCHAR(255), address
VARCHAR(255))")

print("Table Created")

mycursor.execute("SHOW TABLES")
for x in mycursor:
  print(x)




import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
  port = "3306",
  database = "test"
)

print("Connection successful to Database :")

# Insert Into Table

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = ("John", "Highway 21")
mycursor.execute(sql, val)

mydb.commit()


print(mycursor.rowcount, "record inserted.")


# Insert Multiple Rows


sql2 = "INSERT INTO customers (name, address) VALUES (%s, %s)"
```

```
val = [
  ('Peter', 'Lowstreet 4'),
  ('Amy', 'Apple st 652'),
  ('Hannah', 'Mountain 21'),
  ('Michael', 'Valley 345'),
  ('Sandy', 'Ocean blvd 2'),
  ('Betty', 'Green Grass 1'),
  ('Richard', 'Sky st 331'),
  ('Susan', 'One way 98'),
  ('Vicky', 'Yellow Garden 2'),
  ('Ben', 'Park Lane 38'),
  ('William', 'Central st 954'),
  ('Chuck', 'Main Road 989'),
  ('Viola', 'Sideway 1633')
]

mycursor.executemany(sql2, val)

mydb.commit()

print(mycursor.rowcount, "was inserted.")


# Select From a Table

mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
print("\n")

# Selecting Columns

mycursor.execute("SELECT name, address FROM customers")

myresult = mycursor.fetchall()

for x in myresult:
  print(x)


print("\n")
print("\n")
```

```
#   Where
# Select With a Filter

sql = "SELECT * FROM customers WHERE address ='Park Lane 38'"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
  print(x)

# Delete Record
# You can delete records from an existing table by using the "DELETE FROM"
statement:


sql = "DELETE FROM customers WHERE address = 'Mountain 21'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) deleted")


# Update Table
# You can update existing records in a table by using the "UPDATE" statement:

sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley 345'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) affected")
```