// printing program output on the terminal

|  |
| --- |
| print(“hello world”) |

// variables used to temporarily store data in computer memory. For example price, mass, score, and name are variable. Inside the program, variable is first declared by name and then initialized with values

|  |
| --- |
| price = 45 // the assignment operator = is used to assign values to a variable  print(price) |

// types of variables

|  |
| --- |
| Price = 45 // variable takes integer data type  Mass = 12.33 // variable takes float data type  First\_name = “jama” // variable takes string data type  Ice\_is\_white = True // variable types Boolean data type |

// input from user (keyboard)

|  |
| --- |
| Name = input(“what is your first name”) // prompts user to enter first name  Print (“Welcome “ + Name) // concatenates Welcome with first name stored in variable Name |

// data type conversion

|  |
| --- |
| Age = 40  Print(“your age is “ + age) // this concatenation will NOT work  Print (“your age is “ + str(age)) // this will work because integer age is converted to string  Mass = input(“Enter the mass: “) // program return mass as string  Acceleration = 9.8  Force = float(mass)\*acceleration // string mass is converted to float  Print(“force is = “ + str(force)) |

// string data manipulation

|  |
| --- |
| Abstract = “The purpose of this research is to study relationship between smoking and cancer”  Print(abstract) // prints the text in variable abstract  Print(len(abstract)) // counts number of characters in the data  Print(abstract.count(“study”)) // counts how many times study is in the text  Print(abstract.find(“study”)) // index location of the work study  Print(abstract.UPPER()) // converts all text into upper case  Print(abstract.replace(“study”, “STUDY”) // replace study with STUDY |

// arithmetic operators

|  |
| --- |
| Addition +  Subtraction +  Division /  Multiplication \*  Exponent \*\*  Modulus %  Increment +=  Decrement -= |

// comparison operators to compare values. The program outputs Boolean value

|  |
| --- |
| Temperature = 30 > 20 // TRUE  Greater than >  Greater than or equal to >=  Less than <  Less than or equal to <=  Equal to ==  Not equal to != |

// logical operators to return true if certain conditions are met

|  |
| --- |
| AND // returns true if both conditions are true  OR // return true if at least one condition is true  NOT // inverses the condition |

// if condition

|  |
| --- |
| Variable = value  If condition**:**  Execute this code if condition is true // condition uses comparison or logical operator  Example  Raining = True  If raining != True: // This program first sets today as raining. Then the if  Print(“you can go to school”) condition says raining in today in NOT true  Else: then the code in ELSE part is executed  Print(“stay home”)    Ternary if statement used when you have one if and one else  Number = 15  Message = f”{number} is positive” if number >=0 else f”{number} is negative” |

// while loop used to repeat certain codes number of times based on condition

|  |
| --- |
| I = 0 // first initialized the variable  While condition: // condition uses comparison or logical operators  Print()  Increment/decrement  Example  -----------  I = 1 this code will first assigned variable I to 1. The while loop  While 0 < I < 20**:** repeats the code I += 2 from 0 to 20  Print(i) first print output is 1, second print output is 3 after  I += 2 decrementing by 2 |

// list

|  |
| --- |
| Mylist = [1, 2, 3, 4, 5] // data inside list separated by comma  What can we do about this list? 🡺 searching, sorting, replacing, adding, removing, counting  Print(mylist[0]) // will print first element in the list with index of 0 (searching)  Print(mylist[0:3]) // searching range of values  Mylist[0] = -1 // will replace value 1 with -1 (replacing)  Mylist.append(6) // will insert new value 6 at the end of the list (appending)  Mylist.pop() // removes last item in the list  Mylist.insert(1, -2) // will insert new value -2 at the index location 1 (inserting)  Mylist.remove(2) // will remove value 2 from the list (removing)  Mylist.clear() // remove all values in the list (removing all)  Print(2 in mylist) // will check if a value is in the list (finding)  Print(len(mylist)) // how many items are in the list (counting)  Mylist.sort() // sorting the list  Sum all values in the list and break when 4 is reached  sum = 0 for number in numbers:  if number == 4:  break  sum += number print(sum) |

// for loop for iteration

|  |
| --- |
| Mylist = [1, 2, 3, 4, 5]  *For* item *in* mylist:  Print(item)  Names = [“ali”, “hasan”, “ahmed”]  For name in names:  Print(name)  Range function can be used along with for loop to generate sequence of numbers (starting value, ending value, increment)  Numbers = range(0, 20, 3)  For number in numbers:  Print(number) |

// format the print output

|  |
| --- |
| Acceleration = 10  Mass = 40 will output force is 400  Force = mass\*acceleration  Print(f”force is {force}” |

// set is different from list. A set cannot have duplicate values

|  |
| --- |
| set1 = {1, 2, 3, 4, 5} set2 = {-10, 2, 20} union = set1 | set2 // union of two sets  union = set1 & set2 // intersection of two sets print(union) |

// dictionary

|  |
| --- |
| Dictionary = { employee = {    Key : value, “name”: “jama”,  Key : value  “password”: 123  }  }  The following for loop prints items in the  Dictionary  for key, value in employee.items():  print(f"{key} : {value}")  we can get value of specified key in the dictionary and store it in variable  variable = dictionary.get(key) |

// function

|  |
| --- |
| Def <name>(parameter and argument): the following example show a function  Code to execute that takes any number, and then  Adds 3 to that number    def adder():  number = int(input("Enter number = "))  number1 = number + 3  print(number1)   adder() |

// function can take an argument

|  |
| --- |
| def adder(number):  number1 = number + 3  print(number1)   adder(41) |

// function can take more than one argument

|  |
| --- |
| def adder(add, subtract):  add1 = add + 3  subtract1 = subtract - 3  print(f"{add} plus 3 is {add1}")  print(f"{subtract} minus 3 is {subtract1}")   adder(41, 41) |

// function can return value which can then be store in variable

|  |
| --- |
| # write a function that return if a given number if positive or negative  def num\_type(number):  if number >= 0:  return f"{number} is positive"  else:  return f"{number} is negative"   result = num\_type(-10) print(result) |

// import built in function

|  |
| --- |
| Import math // will import built in math function into our program  From math import isqrt // will only import square root from the math function  **import** statistics from statistics **import** mean  mylist = [1, 1, 1] print(statistics.mean(mylist)) mylist = [1, 1, 1]  print(mean(mylist)) |

// creating your own function for re-using in another program

|  |
| --- |
| // adder.py // practice.py    def add(num1, num2): import adder return num1 + num2  print(adder.add(-9, 7)) |

// class and object

|  |
| --- |
| The following class has Department name. It is used to create two objects HR and Finance, each having two **attributes** of director and salary. A **behavior** is added to HR object which tells who many days the HR director works per week  class Department**:**  def \_\_init\_\_(self, director, salary)**:** // this part executed first. Blueprint for  self.director = director all object of the class attributes  self.salary = salary  def workhours(self, work\_hours)**:**  // behavior  print(f"{HR.director} works {work\_hours} days per week")   HR = Department("Ali", 2000) // HR object with passed attributes Finance = Department("yaya", 1000)  print(HR.director) // printing the HR object attribute print(HR.salary) HR.workhours(3) // printing the HR object behavior |

// we can print objects as strings by using the following override method

|  |
| --- |
| def \_\_str\_\_(self):  return f"{self.director}, salary = {self.salary}"  HR = Department("Director of HR is Ali", 2000) Finance = Department("Director of Finance is yaya", 1000)  print(HR) print(Finance) |

// another example of class to calculate the volume of box

|  |
| --- |
| class Box:  def \_\_init\_\_(self, height, length, width):  self.height = height  self.length = length  self.width = width  self.cal\_volume = height\*length\*width   def \_\_str\_\_(self):  return f"The volume of box with {self.height}, " \  f"{self.length}, {self.width} is {self.cal\_volume}"   Volume\_box = Box(3, 3, 3) print(Volume\_box) |

// objects can also be created in list when the number of objects are many

|  |
| --- |
| class Department:  def \_\_init\_\_(self, section, budget):  self.section = section  self.budget = budget   department = [Department("Ali", 2000),  Department("Finance", 1000), Department("Sales", 1500)]  print(department[2].section) |

// import and display current date and time inside program

|  |
| --- |
| from \_datetime import datetime  print(datetime.now()) |

// format date and time using strftime function

|  |
| --- |
| from \_datetime import datetime  Now = datetime.now() // store in variable called Now print(Now.strftime("%d/%m/%Y %H:%M:%S")) |