print("Hello World!")

print(100)

pi = 3.14159

print(pi)

user = "burhanettin"

game = 'chess'

longer = "This string is broken up \

over multiple lines"

user\_name = "busta"

user\_id = 10

verified = False

# A variable's value can be changed after assignment

points = 10

points = 12

age = 7 + 5 = 4

File "<stdin>", line 1

SyntaxError: can't assign to operator

misspelled\_variable\_name

NameError: name 'misspelled\_variable\_name' is not defined

numerator = 100

denominator = 0

bad\_results = numerator / denominator

ZeroDivisionError: division by zero

# Example integer numbers

sofas = 4

tables = 1

broken\_sofas = -2

chairs = 0

# Non-integer numbers

lights = 2.5

left\_overs = 0.0

# Floating point numbers

pi = 3.14159

cost = 12.99

percent = 0.20

# Arithmetic operations

sum = 10 + 30

extraction = 40 - 10

multiply = 50 \* 5

divided = 16 / 4

result = 25 % 2

result = 5 \*\* 3

# Modulo operations

zero = 8 % 4==0

nonzero = 12 % 5==2

# String concatenation

a = "Hello "

b = "World"

c = a + b

c = a + b + "!"

print(a+b+"!")

# Plus-Equal Operator

counter = 0

counter += 10

# This is equivalent to

a1 = 0

a2 = a1 + 10

# The operator will also perform string concatenation

 a1 = "a1 of message "

a2 += "a2 of message"

True or True      # Evaluates to True

True or False     # Evaluates to True

False or False    # Evaluates to False

1 < 2 or 3 < 1    # Evaluates to True

3 < 1 or 1 > 6    # Evaluates to False

1 == 1 or 1 < 2   # Evaluates to True

# elif Statement

if pet\_type == "dog":

  print("You have a dog.")

elif pet\_type == "cat":

  print("You have a cat.")

elif pet\_type == "fish":

  # this is performed

  print("You have a fish")

else:

  print("Not sure!")

 # A 2D list of names and hobbies

hobbies = [["Jenny", "Breakdancing"], ["Alexus", "Photography"], ["Grace", "Soccer"]]

# The sublist of Jenny is at index 0. The hobby is at index 1 of the sublist.

hobbies[0][1] = "Meditation"

print(hobbies)

# Output

# [["Jenny", "Meditation"], ["Alexus", "Photography"], ["Grace", "Soccer"]]

# 2D list of people's heights

weights = [["Hasan", 61], ["Huseyin", 70], ["Ali", 67]]

# Access the sublist at index 0, and then access the 1st index of that sublist.

weights = weights[[0][1], [2][3]=[78,66]

print(weights)

weights = [["Hasan", 78], ["Huseyin", 66], ["Ali", 67]]

# Output

# 61

# Create a list

shopping\_list = ["Cole", "Kip", "Chris", "Sylvana", "Chris"]

# Removes the first occurance of "Chris"

shopping\_list.remove("Chris")

print(shopping\_list)

# Output

# ["Cole", "Kip", "Sylvana", "Chris"]

backpack = ['pencil', 'pen', 'notebook', 'textbook', 'pen', 'highlighter', 'pen']

numPen = backpack.count('pen')

print(numPen)

# Output: 3

list = [2, 4, 3, 7, 10]

size = len(list)

print(size)

# Output: 5

List = [4, 2, 1, 3]

List.sort()

print(List)

# Output: [1, 2, 3, 4]

tools = ['pen', 'hammer', 'lever']

slice = tools[1:3] # ['hammer', 'lever']

slice[0] = 'nail'

# Original list is unaltered:

print(tools) # ['pen', 'hammer', 'lever']

unList = [4, 2, 1, 3]

sList = sorted(List)

print(sList)

# Output: [1, 2, 3, 4]

# Here is a list representing a line of people at a store

list = ["Karla", "Maxium", "Martim", "Isabella"]

# Here is how to insert "Vikor" after "Maxium" and before "Martim"

list.insert(2, "Vikor")

print(list)

# Output: ['Karla', 'Maxium', 'Vikor', 'Martim', 'Isabella']

topics = ["Python", "Data Structures", "Balloon Making", "Algorithms", "Clowns 101"]

# Pop the last element

removed\_element = topics.pop()

print(topics)

print(removed\_element)

# Output:

# ['Python', 'Data Structures', 'Balloon Making', 'Algorithms']

# 'Clowns 101'

# Pop the element "Baloon Making"

topics.pop(2)

print(topics)

# Output:

# ['Python', 'Data Structures', 'Algorithms']

numbers = [0, 254, 2, -1, 3]

for num in numbers:

  if (num < 0):

    print("Negative number detected!")

    break

  print(num)

# 0

# 254

# 2

# Negative number detected!

# List comprehension for the squares of all even numbers between 0 and 9

list\_comprehension = [x\*\*2 for x in range(10) if x % 2 == 0]

print(List\_comprehension)

# [0, 4, 16, 36, 64]

for <temporary variable> in <list variable>:

  <action statement>

  <action statement>

#each num in nums will be printed below

nums = [1,2,3,4,5]

for num in nums:

  print(num)

big\_number\_list = [1, 2, -1, 4, -5, 5, 2, -9]

# Print only positive numbers:

for i in big\_number\_list:

  if i < 0:

    continue

  print(i)

  # Print the numbers 0, 1, 2:

for i in range(3):

  print(i)

# Print "WARNING" 3 times:

for i in range(3):

  print("WARNING")

  # This loop will only run 1 time

hungry = True

while hungry:

  print("Time to eat!")

  hungry = False

# This loop will run 5 times

i = 1

while i < 6:

  print(i)

  i = i + 1

  groups = [["Jobs", "Gates"], ["Newton", "Euclid"], ["Einstein", "Feynman"]]

# This outer loop will iterate over each list in the groups list

for group in groups:

  # This inner loop will go through each name in each list

  for name in group:

    print(name)

    umbers = [0, 254, 2, -1, 3]

for num in numbers:

  if (num < 0):

    print("Negative number detected!")

    break

  print(num)

# 0

# 254

# 2

# Negative number detected!

a = 5

def f1():

  a = 2

  print(a)

print(a)   # Will print 5

f1()       # Will print 2

def write\_a\_book(character, setting, special\_skill):

  print(character + " is in " +

        setting + " practicing her " +

        special\_skill)

ef ready\_for\_school(backpack, pencil\_case):

  if (backpack == 'full' and pencil\_case == 'full'):

    print ("I'm ready for school!")

def check\_leap\_year(year):

  if year % 4 == 0:

    return str(year) + " is a leap year."

  else:

    return str(year) + " is not a leap year."

year\_to\_check = 2018

returned\_value = check\_leap\_year(year\_to\_check)

print(returned\_value) # 2018 is not a leap year.

# Define a function my\_function() with parameter x

def my\_function(x):

  return x + 1

# Invoke the function

print(my\_function(2))      # Output: 3

print(my\_function(3 + 5))  # Output: 9

# Indentation is used to identify code blocks

def testfunction(number):

  # This code is part of testfunction

  print("Inside the testfunction")

  sum = 0

  for x in range(number):

    # More indentation because 'for' has a code block

    # but still part of he function

    sum += x

  return sum

print("This is not part of testfunction")

a = "Hello World!"

def prints\_a():

  print(a)

# will print "Hello World!"

prints\_a()

def my\_function(value):

  print(value)

# Pass the value 7 into the function

my\_function(7)

# Causes an error as `value` no longer exists

print(value)

def sales(grocery\_store, item\_on\_sale, cost):

  print(grocery\_store + " is selling " + item\_on\_sale + " for " + cost)

sales("The Farmer’s Market", "toothpaste", "$1")

def findvolume(length=1, width=1, depth=1):

  print("Length = " + str(length))

  print("Width = " + str(width))

  print("Depth = " + str(depth))

  return length \* width \* depth;

findvolume(1, 2, 3)

findvolume(length=5, depth=2, width=4)

findvolume(2, depth=3, width=4)