|  |  |
| --- | --- |
| **DOCUMENT RULES:** | |
| **Task Number / Name:** | **Python** |
| **Task name & column name should be written:** | **Bold (CTRL+B)** |
| **Commands should be written in the after # sign:** | *Italic (CTRL+I) #hostname* |
| **Output photo should be cropped or compressed:**  **Photo could be more than one:**  **If you need extra lines, add the line next after it:** | ***Description photo should be with title bar (CTRL + I + B)*** |
| **All other text should be written:** | Standard |
| **Font name and text size:** | Calibri and 9 |
| **Group name:** | Dev\_ops\_ |
| **Student name and surname:** | Murad Abbaszade |
| **E-mail:** | [muradabbaszade6@gmail.com](mailto:muradabbaszade6@gmail.com) |
| **WhatsApp number:** | **+994703664205** |

Python

print(a)

number = input(“Print number:”)

If else elif statement

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")

Loops

While

largest\_number = -999999999

number = int(input("Enter a number or type -1 to stop: "))

while number != -1:

if number > largest\_number:

largest\_number = number

number = int(input("Enter a number or type -1 to stop: "))

print("The largest number is:", largest\_number)

For break continue

fruits = ["apple", "banana", "cherry"]

for x in fruits:

print(x)

if x == "banana":

break;

continue

fruits = ["apple", "banana", "cherry"]

for x in fruits:

if x == "banana":

continue

print(x)

Lists--array

myNumbers = [10,5,7,2,1]

Print(myNumbers)

len(myNumbers)-length

del myNumbers[1]- delete element

print(myNumbers[-1]) # last number

print(myNumbers[-2]) # second to last

fruits.append("orange") – Add an element to the list.

fruits.insert(1, "orange") - Insert new value as the second element of the list.

fruits.pop(1) - Remove the second element of the list.

fruits.remove("banana")- Remove a specific element of the list.

Dictionaries

thisdict = {

"brand": "Ford",

"model": "Mustang",

"year": 1964

}

thisdict = {

"brand": "Ford",

"electric": False,

"year": 1964,

"colors": ["red", "white", "blue"]

}

print(thisdict[“colors”])

Functions

def my\_function(fname, lname):

print(fname + " " + lname)

def- define function

my\_function("Emil", "Refsnes")

Import

import time

print("this will print immediately")

time.sleep(3)

print("this printed after a 3 second pause")

\*\*\*\*\*\*

import datetime

x = datetime.datetime.now()

print(x)

\*\*\*\*\*\*

import datetime as myCoolModule

x = myCoolModule.datetime.now()

print(x)

\*\*\*\*\*\*

(1)def greeting(name):

print("Hello, " + name)

(2)import myPersonalModule

myPersonalModule.greeting("Jonathan")

1 ve 2 eynidir.

\*\*\*\*\*\*

# Creating a function called my\_function()

def print\_tuple(\*friends):

# The function will then print each name

for friend in friends:

print(friend)

# setting up the tuple

friends = ("Aymen", "Emma", "Sandra", "Yasmin")

# calling the function

print\_tuple(friends)

Classes in Python

class className():

name = "default"

number = 0

class className():

def sum(self, int1, int2):

return int1 + int2

objectName = className(value1, value2, value3...) # 'values' will set the variables for that object to those values

\*\*\*\*\*\*

class Calculator(object):

total = 0

# the variable values established when creating the object

def \_\_init\_\_(self, var1, var2): #constructor

# assigning the input values to the class variables

self.int1 = var1

self.int2 = var2

# creating an object from the class

example = Calculator(24, 391)

# calling the 'sum' method and printing the results

print("Storage: Value 1 = ", example.int1, ", Value 2 = ", example.int2)

\*\*\*\*\*\*

objectName = className(value1, value2, value3...) # sets values to attributes

objectName.methodName() # executes method

\*\*\*\*\*\*

class Calculator(object):

total = 0

int1 = 0

int2 = 0

# creating an object from the class

example = Calculator()

example.int1 = 24

example.int2 = 391

Inheritance in Python

class Vehical(object): # Parent/Super class

type = "vehical"

def \_\_init\_\_(self, seats, colouring):

self.nb\_of\_seats = seats

self.colour = colouring

def specs(self):

print("I am a " + self.type + ". I have " + str(self.nb\_of\_seats) + " seats and I am " + self.colour)

class Car(Vehical): # Child/Sub class to Parent

type = "car"

def description(self):

print("This description is only for cars")

vehical = Vehical(4, "red")

car = Car(2, "blue")

vehical.specs()

car.description()

car.specs()

class Car(object):

def \_\_init\_\_(self, model, passengers, colour, speed):

self.car\_type = model

self.max\_passengers = passengers

self.colour = colour

self.speed = speed

def get\_car\_data(self):

list = [self.car\_type, self.max\_passengers, self.colour, self.speed]

return list

def calculate\_acceleration(self):

self.speed = self.speed + 2

print (self.speed)

C = Car("BMW", 4, "purple", 50)

print(C.get\_car\_data())

C.calculate\_acceleration()