|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| C:\Users\Admin\Desktop\download.jpg | USMAN INSTITUTE OF TECHNOLOGY | | | | | |  |
|  |  |  |  |  |  |  |  |
|  | Department of Computer Science  CS121 Object Oriented Programming | | | | | |  |
|  |  | Lab # 02  Elementary Programming in Python | | | |  |  |
|  | Objective:  This experiment introduces the students to Elementary concepts of Programming in Python programming language. The use of variables is explored. Python Data Types and various operations are introduced. Evaluating Expressions and use of Assignment Statement is practiced | | | | | |  |
|  | **Name of Student:**  **Roll No: Sec.**  **Date of Experiment:** | | | | | |  |
|  | **Marks Obtained/Remarks:**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Signature:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | |  |

**Lab 02: Elementary Programming In Python**

# Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable.

counter = 100 # An integer assignment

miles = 1000.0 # A floating point

name = "Usman" # A string

print (counter)

print (miles); print (name)

# Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types

* Numbers
* String
* List
* Tuple
* Dictionary

# Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them.

var1 = 1

var2 = 10

# Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:] ) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

str = 'Hello World!'

print (str) # Prints complete string

print (str[0]) # Prints first character of the string

print (str[2:5]) # Prints characters starting from 3rd to 5th

print (str[2:]) # Prints string starting from 3rd character

print (str \* 2) # Prints string two times

print (str + "TEST") # Prints concatenated string

This will produce the following result

Hello World!

H

Llo

llo World!

Hello World!Hello World!

Hello World!TEST

# Python Lists

The list is the most versatile data type available in Python, which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that the items in a list need not be of the same type. Creating a list is as simple as putting different comma-separated values between square brackets.

list1 = ['physics', 'chemistry', 1997, 2000]

list2 = [1, 2, 3, 4, 5, 6, 7 ]

print ("list1[0]: ", list1[0])

print ("list2[1:5]: ", list2[1:5])

When the above code is executed, it produces the following result

list1[0]: physics

list2[1:5]: [2, 3, 4, 5]

# Updating Lists

You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator, and you can add to elements in a list with the append() method.

list = ['physics', 'chemistry', 1997, 2000]

print ("Value available at index 2 : ", list[2])

list[2] = 2001

print ("New value available at index 2 : ", list[2])

# Delete List Elements

To remove a list element, you can use either the del statement if you know exactly which element(s) you are deleting. You can use the remove() method if you do not know exactly what is the index of the item to delete.

list = ['physics', 'chemistry', 1997, 2000]

print (list)

del list[2]

print ("After deleting value at index 2 : ", list)

# Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parenthesis. The main difference between lists and tuples are − Lists are enclosed in brackets ( [ ] ) and their elements and size can be changed, while tuples are enclosed in parentheses ( ( ) ) and cannot be updated. Tuples can be thought of as read-only lists.

tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )

tinytuple = (123, 'john')

print (tuple) # Prints complete tuple

print (tuple[0]) # Prints first element of the tuple

print (tuple[1:3]) # Prints elements starting from 2nd till 3rd

print (tuple[2:]) # Prints elements starting from 3rd element

print (tinytuple \* 2) # Prints tuple two times

print (tuple + tinytuple) # Prints concatenated tuple

This produces the following result

('abcd', 786, 2.23, 'john', 70.200000000000003)

abcd

(786, 2.23)

(2.23, 'john', 70.200000000000003)

(123, 'john', 123, 'john')

('abcd', 786, 2.23, 'john', 70.200000000000003, 123, 'john')

The following code is invalid with tuple, because we attempted to update a tuple, which is not allowed. Similar case is possible with lists

tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )

list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]

tuple[2] = 1000 # Invalid syntax with tuple

list[2] = 1000 # Valid syntax with list

# Python Dictionary

Python's dictionaries are kind of hash-table type. They consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object. Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

dict = {}

dict['one'] = "This is one"

dict[2] = "This is two"

tinydict = {'name': 'john','code':6734, 'dept': 'sales'}

print (dict['one']) # Prints value for 'one' key

print (dict[2]) # Prints value for 2 key

print (tinydict) # Prints complete dictionary

print (tinydict.keys()) # Prints all the keys

print (tinydict.values()) # Prints all the values

This produces the following result

This is one

This is two

{'name': 'john', 'dept': 'sales', 'code': 6734}

dict\_keys(['name', 'dept', 'code'])

dict\_values(['john', 'sales', 6734])

# Student Exercise

1. Write a program that reads in the radius and length of a cylinder and computes the area and volume using the following formula

*area = radius \* radius \* π*

*volume = area \* length*

***Code:*** #Task 1

from math import \*

#input

length = int(input("Length"))

radius = int(input("Radius"))

#formula

area = radius \* radius \* pi

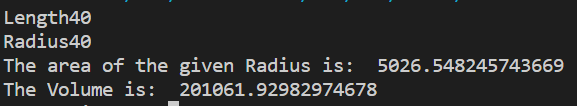
volume = area \* length

#print

print("The area of the given Radius is: ",area)

print("The Volume is: ",volume)

**OUTPUT:**

******

1. Write a program that prompts the user to enter the minutes (e.g. 1 billion) and displays the number of years and days for the minutes. For simplicity, assume a year has 365 days.

***Code:*** #Task 2

#input

minutes=eval(input("enter minutes: "))

#Converting

years =minutes / 525600

days =(minutes % 525600) / 1440

#print

print(minutes , " minutes is approximately " , years, " years and " , days , " days")

**OUTPUT:**



1. Write a program that calculates the energy needed to heat water from an initial temperature to a final temperature. Your program should prompt the user to enter the amount of water in kilograms and the initial and final temperatures of the water. The formula to compute the energy is

*Q = M \* (finalTemperature – initialTemperature ) \* 4184*

Where M is the weight of water in kilograms, temperatures are in degrees Celsius, and energy Q is measured in joules.

***Code:*** #Task 3

M = int(input("Enter amount of water in kilograms "))

finalTemperature = int(input("Enter Final Temperature in degrees Celsius: "))

initialTemperature = int(input("Enter Initial Temperature in degrees Celsius: "))

#Compute

Q = M \* (finalTemperature - initialTemperature ) \* 4184

#print

print("energy in joules: ",Q)

**OUTPUT:**

***Text

Description automatically generated***

1. Write a program that prompts the user to enter a four-digit number and displays the number in reverse order

***Code:*** # Task 4

n = int(input("Enter a 4 digit integer: "))

reverse = 0

while (n > 0):

    a = n % 10

    reverse = reverse \* 10 + a

    n = n // 10

print(reverse)

**OUTPUT:**

******

1. Body Mass Index (BMI) is a measure of health based on weight. It can be calculated by taking your weight in kilograms and dividing it by the square of your height in meters. Write a program that prompts the user to enter a weight in pounds and height in inches and displays the BMI.

***Code:*** # Task 5

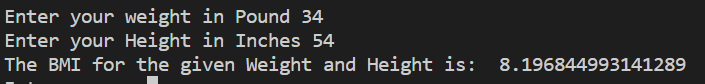
Weight = int(input("Enter your weight in Pound "))

Height = int(input("Enter your Height in Inches "))

BMI = (Weight / Height\*\*2) \* 703

print("The BMI for the given Weight and Height is: ",BMI)

**OUTPUT:**

******

1. The US Census Bureau projects population based on the following assumptions
   1. One birth every 7 seconds
   2. One death every 13 seconds
   3. One immigrant every 45 seconds

Write a program to prompt the user to enter the number of years and displays the population after that many years.

***Code:*** # Task 6

no\_of\_seconds\_in\_year = 365\*24\*60\*60

birthsperyear = no\_of\_seconds\_in\_year/7

deathsperyear = no\_of\_seconds\_in\_year/13

immigrantsperyear = (no\_of\_seconds\_in\_year/45)

x= int(input("Enter an year"))

population\_rate = x\*(birthsperyear + deathsperyear - immigrantsperyear)

print(birthsperyear, int(birthsperyear) , 'are the number of births per year')

print(deathsperyear, int(deathsperyear), 'are the number of deaths per year')

print(immigrantsperyear, int(immigrantsperyear), 'are the number of immigrants per year')

print(population\_rate, 'population to that year')

**OUTPUT:**

Text

Description automatically generated