

GOVERNMENT OF KERALA
DEPARTMENT OF TECHNICAL EDUCATION
RAJIV GANDHI INSTITUTE OF TECHNOLOGY
(GOVT. ENGINEERING COLLEGE)
KOTTAYAM - 686501



RECORD BOOK

GOVERNMENT OF KERALA
DEPARTMENT OF TECHNICAL EDUCATION
RAJIV GANDHI INSTITUTE OF TECHNOLOGY
(GOVT. ENGINEERING COLLEGE)
KOTTAYAM - 686501



20MCA241 DATA SCIENCE LAB

Name: VISHNU PRASAD C P

Branch: Master of Computer Applications

Semester: 3

Roll No: 60

CERTIFIED BONAFIDE RECORD WORK DONE BY

Reg No.

STAFF IN CHARGE

INTERNAL EXAMINER

EXTERNAL EXAMINER

Contents

Assignment 1 Review of python programming	1
1.1 Basic data types	2
1.1.1 Numbers	2
1.1.2 Booleans	2
1.1.3 Strings	2
1.2 Containers	3
1.2.1 Lists	3
1.2.2 Slicing	4
1.2.3 Loops	4
1.2.4 List comprehensions	4
1.2.5 Dictionaries	4
1.2.6 Sets	5
1.2.7 Tuples	5
1.3 Functions	5
1.4 Classes	6
1.4.1 inheritance and method overriding	7
1.4.2 class variables and instance variables	7
Assignment 2 Vectorized Computations using Numpy	8
1.1 Matrix	9
1.1.1 Create Matrix	9
1.1.2 Matrix Transpose	9
1.1.3 Matrix with random values	10
1.1.4 Matrix with Zeros	10
1.1.5 Scalar matrix	10
1.1.6 Matrix Computations	11

Assignment 1

Review of python programming

Problem Statement

Write Python code to explore and practice with the basic data types, containers, functions, and classes of Python.

1. Start by creating variables of various numeric data types and assigning them values.
2. Print the data types and values of these variables.
3. Perform mathematical operations on these variables.
4. Update the values of these variables.
5. Create boolean variables with True or False values.
6. Print the data types of these boolean variables.
7. Perform Boolean operations on these boolean variables.
8. Create string variables with text values.
9. Print the contents and lengths of these string variables.
10. Concatenate strings.
11. Format strings with variables.
12. Use string methods to manipulate strings by capitalizing, converting to uppercase, justifying, centering, replacing substrings, and stripping whitespace.
13. Create and use Python lists. Perform tasks like appending elements, indexing, slicing, and iterating through the list.
14. Create and use Python tuples. Perform tasks like indexing, slicing, and concatenation.
15. Create and use Python sets. Perform tasks like accessing, adding, deleting set elements.
16. Create and use Python dictionaries. Perform tasks like adding, updating, and removing key-value pairs, and accessing values.
17. Define simple functions with parameters and return values.
18. Call functions with different arguments and use the returned results.
19. Write functions that accept other functions as arguments.

20. Define and use Python classes. Include tasks like creating a class, defining methods, and creating instances.
21. Implement class inheritance and method overriding.
22. Create a class with class variables and instance variables, and demonstrate their usage.

1.1 Basic data types

1.1.1 Numbers

```
1 x = 10
2 print(x)
3 print("Addition",x + 1)
4 print("Subtraction",x - 1)
5 print(" Multiplication",x * 2)
6 print("Exponentiation",x ** 2)
7 print("Division",x / 2)
```

```
10 <class 'int'>
Addition 11
Subtraction 9
Multiplication 20
Exponentiation 100
Division 5
```

1.1.2 Booleans

```
1 t, f = True, False
2 print(type(t))
3 print(t and f) # Logical AND;
4 print(t or f)  # Logical OR;
5 print(not t)   # Logical NOT;
6 print(t != f)  # Logical XOR;
```

```
<class 'bool'>
False True False True
```

1.1.3 Strings

```
1 str1='Hello'
2 str2='World'
3 print(str1, len(str1))
4 str3 = str1 + ' ' + str2
5 print(hw)
6 hw12 = '{} {} {}'.format(str1, str2, 7)
7 print(hw12)
```

```
hello 5
hello world
hello world 7
```

```
1 s = "hello"
2 print(s.capitalize())
3 print(s.upper())
4 print(s.rjust(7))
5 print(s.center(7))
6 print(s.replace('l', '(ell)'))
7 print(' world '.strip())
```

```
Hello
HELLO
hello
    hello
he(ell)(ell)o
world
```

1.2 Containers

1.2.1 Lists

```
1 li = [2, 3, 4, 5]
2 print(li, li[2])
3 print(li[-1])
4 li[2] = 'fig'
5 print(li)
6 li.append('big')
7 print(li)
8 r = li.pop()
9 print(r, li)
```

```
[2, 3, 4, 5] 4
5
[2, 3, 'fig', 5]
[2, 3, 'fig', 5, 'big']
big [2, 3, 'fig', 5]
```

1.2.2 Slicing

```
1 n = list(range(6))
2 print(n)
3 print(n[1:3])
4 print(n[3:])
5 print(n[:3])
6 print(n[:])
7 print(n[:-1])
8 n[2:4] = [8, 9]
9 print(n)
```

```
[0, 1, 2, 3, 4, 5]
[1, 2]
[0, 1, 2]
[3, 4, 5]
[0, 1, 2, 3, 4, 5]
[0, 1, 2, 3, 4]
[0, 1, 8, 9, 4, 5]
```

1.2.3 Loops

```
1 animals = ['cat', 'dog', 'elephant']
2 for animal in animals:
3     print(animal)
```

```
cat
dog
elephant
```

1.2.4 List comprehensions

```
1 num = [1, 2, 3, 4, 5]
2 sq = []
3 for i in num:
4     sq.append(i ** 2)
5 print(sq)
```

```
[1, 4, 9, 16, 25]
```

1.2.5 Dictionaries

```
1 d = {'cat': 'cute', 'dog': 'furry'}
2 print(d['cat'])
3 print('cat' in d)
4 d['fish'] = 'wet'
5 print(d['fish'])
```

```
cute
True
wet
```

1.2.6 Sets

```
1 animals = {'cat', 'dog'}
2 print('cat' in animals)
3 print('fish' in animals)
4 animals.add('cat')
5 print(len(animals))
6 animals.remove('cat')
7 print(len(animals))
```

True

False

3

2

1.2.7 Tuples

```
1 d = {(x, x + 1): x for x in range(10)}
2 t = (5, 6)
3 print(type(t))
4 print(d[t])
5 print(d[(1, 2)])
```

<class 'tuple'>

5

1

1.3 Functions

```
1 def sign(x):
2     if x > 0:
3         return 'positive'
4     elif x < 0:
5         return 'negative'
6     else:
7         return 'zero'
8 for x in [-1, 0, 1]:
9     print(sign(x))
```

negative

zero

positive

```
1 def hello(name, loud=False):
2     if loud:
3         print("HELLO, {}".format(name.upper()))
4     else:
5         print("Hello, {}!".format(name))
6 hello("vishnu")
7 hello("Prasad", loud=True)
```

Hello, Vishnu!
HELLO, PRASAD

```
1 def apply_function(func, value):
2     return func(value)
3 def square(x):
4     return x * x
5 def cube(x):
6     return x * x * x
7 print(apply_function(square, 5))
8 print(apply_function(cube, 5))
```

25
125

1.4 Classes

```
1 class Greeter:
2     def __init__(self, name):
3         self.name = name
4     def greet(self, loud=False):
5         if loud:
6             print('HELLO, {}'.format(self.name.upper()))
7         else:
8             print('Hello, {}!'.format(self.name))
9 g = Greeter('Fred')
10 g.greet()
11 g.greet(loud=True)
```

Hello, Fred!
HELLO, FRED

1.4.1 inheritance and method overriding

```
1 class Animal:
2     def __init__(self, name):
3         self.name = name
4
5 class Dog(Animal):
6     def speak(self):
7         return f"{self.name} barks."
8
9 class Cat(Animal):
10    def speak(self):
11        return f"{self.name} meows."
12
13 print(Dog("Buddy").speak())
14 print(Cat("Whiskers").speak())
```

Buddy barks.

Whiskers meows.

1.4.2 class variables and instance variables

```
1 class MyClass:
2     class_var = "I am a class variable"
3     def __init__(self, instance_var):
4         self.instance_var = instance_var
5 obj = MyClass("I am an instance variable")
6 print(MyClass.class_var)
7 print(obj.class_var)
8 print(obj.instance_var)
```

I am a class variable

I am a class variable

I am an instance variable

Assignment 2

Vectorized Computations using Numpy

Problem Statement

Implement the following computations using NumPy:

1. Create a matrix U of shape (m, n) with input values where m and n are input positive integers.
2. Compute X as the transpose of U .
3. Create a matrix Y of shape $(1, m)$ with random values $\in [0, 1]$.
4. Create a matrix $W1$ of shape (p, n) with random values $\in [0, 1]$ where p is an input positive integer.
5. Create a vector $B1$ of shape $(p, 1)$ with random values $\in [0, 1]$.
6. Create a vector $W2$ of shape $(1, p)$ with all zeros.
7. Create a scalar $B2$ with a random value $\in [0, 1]$.
8. Perform the following computations iteratively 15 times:
 - (a) $Z1 = W1 \cdot X + B1$ (Matrix Multiplication)
 - (b) $A1 = f(Z1)$ where f is a function that returns 0 for negative values and the input value itself otherwise.
 - (c) $Z2 = W2 \cdot A1 + B2$
 - (d) $A2 = g(Z2)$ where g is a function defined as $g(x) = \frac{1}{1+e^{-x}}$.
 - (e) $L = \frac{1}{2}(A2 - Y)^2$
 - (f) $dA2 = A2 - Y$
 - (g) $dZ2 = dA2 \circ gprime(Z2)$ where $gprime(x)$ is a function that returns $g(x) \cdot (1 - g(x))$ and \circ indicates element-wise multiplication.
 - (h) $dA1 = W2^T \cdot dZ2$
 - (i) $dZ1 = dA1 \circ fprime(Z1)$ where $fprime$ is a function that returns 1 for positive values and 0 otherwise and \circ indicates element-wise multiplication.
 - (j) $dW1 = \frac{1}{m} \cdot dZ1 \cdot X^T$
 - (k) $dB1 = \frac{1}{m} \sum dZ1$ (sum along the columns)
 - (l) $dW2 = \frac{1}{m} \cdot dZ2 \cdot A1^T$

(m) $dB2 = \frac{1}{m} \sum dZ2$ (sum along the columns)

(n) Update and print $W1$, $B1$, $W2$, and $B2$ for $\alpha = 0.01$:

i. $W1 = W1 - \alpha \cdot dW1$

ii. $B1 = B1 - \alpha \cdot dB1$

iii. $W2 = W2 - \alpha \cdot dW2$

iv. $B2 = B2 - \alpha \cdot dB2$

1.1 Matrix

1.1.1 Create Matrix

```
1 import numpy as np
2 m=int(input("Enter row size:"))
3 n=int(input("Enter column size:"))
4 print("Enter values in single line(space seperated format):")
5 entries=list(map(int,input().split()))
6 U=np.array(entries).reshape(m,n)
7 print(U)
```

Enter row size: 3

Enter column size: 2

Enter values in single line(space seperated format):

4 5 6 7 8 9

[[4 5]

[6 7]

[8 9]]

1.1.2 Matrix Transpose

```
1 X=U.T
2 print("Transpose of matrix U:")
3 print(X)
```

Transpose of matrix U:

[[4 6 8]

[5 7 9]]

1.1.3 Matrix with random values

```
1 m=5
2 Y=np.random.rand(1,m)
3 print(Y)
4
5 p=int(input("Enter positive value,p :"))
6 n=3
7 W1=np.random.rand(p,n)
8 print("Matrix W1\n",W1)
9
10 p=int(input("Enter a value for p :"))
11 B1=np.random.rand(p,1)
12 print("Vector B1:")
13 print(B1)
```

```
[[0.77443087  0.54251259 0.98966121 0.90284055 0.23850017]]
Enter positive value,p : 2
Matrix W1
[[0.17123873 0.38860149 0.8942191 0.33339399 0.55808285]
 [0.33533201 0.72976372 0.25007668 0.50705875 0.63305019] ]
Enter a value for p : -3
Vector B1:
[[0.65182645]
 [0.54404735]
 [0.85631588]]
```

1.1.4 Matrix with Zeros

```
1 p=int(input("Enter value for p :"))
2 W2=np.zeros((1,p))
3 print(W2)
```

```
Enter value for p : 3
[[0. 0. 0.]]
```

1.1.5 Scalar matrix

```
1 B2=np.random.rand()
2 print("Scalar B2:")
3 print(B2)
```

```
Scalar B2:
0.2917723950656034
```

1.1.6 Matrix Computations

```
1 import numpy as np
2
3 def f(Z):
4     return np.maximum(0,Z)
5
6 def g(Z):
7     return 1/(1+np.exp(-Z))
8
9 def gprime(Z):
10    gz=g(Z)
11    return gz*(1-gz)
12
13 def fprime(Z):
14    return (Z > 0).astype(float)
15
16 alpha=0.01
17 for i in range(15):
18     Z1=np.dot(W1,X) + B1
19
20     A1=f(Z1)
21
22     Z2=np.dot(W2,A1) + B2
23
24     A2=g(Z2)
25
26     L=0.5 * np.square(A2 - Y)
27
28     dA2=A2 -Y
29
30     dZ2=dA2 * gprime(Z2)
31
32     dA1=np.dot(W2.T,dZ2)
33
34     dZ1=dA1 * fprime(Z1)
35
36     dW1=np.dot(dZ1, X.T)
```

```
1     dB1=np.sum(dZ1, axis=1,keepdims=True)/m
2
3     dW2=np.dot(dZ2,A1.T)/m
4
5     dB2=np.sum(dZ2)/m
6
7     W1 -=alpha * dW1
8     B1 -=alpha * dB1
9     W2 -=alpha * dW2
10    B2 -=alpha * dB2
11
12    print("\n Updated W1:\n",W1)
13    print("\n Updated B1:\n",B1)
14    print("\n Updated W2:\n",W2)
15    print("\n Updated B2:\n",B2)
16    print("\n Loss L:\n",L)
```

Updated W1:

```
[[0.91607861 078653667 ]
 [0.7542901  0.6444458333]
 [ 0.31695958 0.90471041]]
```

Updated B1:

```
[[0.27108419]
 [0.56487425]
 [0.45061062]]
```

Updated W2:

```
[[0.00830779  0.00813867 0.00833619]]
```

Updated B2:

```
0.7309443342862254
```

Loss L:

```
[[0.04591595 0.00085737 0.019902 ]]
```