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

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

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

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

1.1.2 Booleans

```
t, f = True, False
print(type(t))
print(t and f) # Logical AND;
print(t or f)  # Logical OR;
print(not t)   # Logical NOT;
print(t != f)  # Logical XOR;
```

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

1.1.3 Strings

```
    str1='Hello'
    str2='World'
    print(str1, len(str1))
    str3 = str1 + ' ' + str2
    print(hw)
    hw12 = '{} {} {}'.format(str1, str2, 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

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

```
animals = ['cat', 'dog', 'elephent']
for animal in animals:
    print(animal)
```

```
cat
dog
elephent
```

1.2.4 List comprehensions

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

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

1.2.5 Dictionaries

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

```
cute
True
wet
```

1.2.6 Sets

```
animals = {'cat', 'dog'}
print('cat' in animals)
print('fish' in animals)
animals.add('cat')
print(len(animals))
animals.remove('cat')
print(len(animals))
```

```
True
False
3
2
```

1.2.7 Tuples

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

```
<class 'tuple'>
5
1
```

1.3 Functions

```
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'
for x in [-1, 0, 1]:
    print(sign(x))
```

```
negative
zero
positive
```

```
def hello(name, loud=False):
    if loud:
        print("HELLO, {}".format(name.upper()))
    else:
         print("Hello, {}!".format(name))
hello("vishnu")
hello("Prasad", loud=True)
```

```
Hello, Vishnu!
HELLO, PRASAD
```

```
def apply_function(func, value):
    return func(value)
def square(x):
    return x * x
def cube(x):
    return x * x * x
print(apply_function(square, 5))
print(apply_function(cube, 5))
```

```
25
125
```

1.4 Classes

```
class Greeter:
    def __init__(self, name):
        self.name = name
    def greet(self, loud=False):
        if loud:
          print('HELLO, {}'.format(self.name.upper()))
        else:
          print('Hello, {}!'.format(self.name))
g = Greeter('Fred')
g.greet()
g.greet(loud=True)
```

```
Hello, Fred!
HELLO, FRED
```

1.4.1 inheritance and method overriding

```
class Animal:
    def __init__(self, name):
        self.name = name

class Dog(Animal):
    def speak(self):
        return f"{self.name} barks."

class Cat(Animal):
    def speak(self):
        return f"{self.name} meows."

print(Dog("Buddy").speak())
print(Cat("Whiskers").speak())
```

```
Buddy barks.
Whiskers meows.
```

1.4.2 class variables and instance variables

```
class MyClass:
    class_var = "I am a class variable"
    def __init__(self, instance_var):
        self.instance_var = instance_var
obj = MyClass("I am an instance variable")
print(MyClass.class_var)
print(obj.class_var)
print(obj.instance_var)
```

```
I am a class variable
I am a class variable
I am an instance variable
```