Assignment_1

July 18, 2025

0.1 Assignment 1

1 Review of Python Programming

1. Start by creating variables of various numeric data types and assigning them values.

[2]:
```
a=19
b=20.56
c=4+9j
d='anu'
```

2. Print the data types and values of these variables.

[3]:
```
print(type(a),a)
print(type(b),b)
print(type(c),c)
print(type(d),d)
```

```
<class 'int'> 19
<class 'float'> 20.56
<class 'complex'> (4+9j)
<class 'str'> anu
```

3. Perform mathematical operations on these variables

[6]:
```
print("sum :",a+b)
print("difference",a-b)
print("product",a*c)
```

```
sum : 39.56
difference -1.5599999999999987
product (76+171j)
```

4. Update the values of these variables.

[9]:
```
a=a+5
b=b-2
print("update a:",a)
print("update b:",b)
```

```
update a: 24
update b: 18.56
```

5. Create boolean variables with True or False values.

[13]:
```
f1=True
f2=False
```

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

6. Print the data types of these boolean variables.

[14]:
```
print(type(f1),f1)
print(type(f2),f2)
```

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

7. Perform Boolean operations on these boolean variables.

[16]:
```
print("AND operation :",f1 and f2)
print("OR operation :",f1 or f2)
print("NOT operation:",not f2)
```

```
AND operation : False
OR operation : True
NOT operation: True
```

8. Create string variables with text values.

[17]:
```
str1="hello"
str2="world"
```

9. Print the contents and lengths of these string variables.

[18]:
```
print("str1:",str1,"| Length:",len(str1))
print("str2:",str2,"| Length:",len(str2))
```

```
str1: hello | Length: 5
str2: world | Length: 5
```

10. Concatenate strings.

[19]:
```
concat=str1+" "+str2
print("Concated String:",concat)
```

```
Concated String: hello world
```

11. Format strings with variables.

[20]:
```
formatted_str=f"{str1.upper()} {str2.capitalize()}"
print("Formatted String :",formatted_str)
```

```
Formatted String : HELLO World
```

12. Use string methods to manipulate strings by capitalizing, converting to uppercase, justifying, centering, replacing substrings, and stripping whitespace.

[22]:
```
print("Capitalized:",str1.capitalize())
print("Uppercase:",str2.upper())
print("Right justified:",concat.rjust(20))
print("Centered:",concat.center(20))
print("Replaced:",concat.replace("world","python"))
print("Stripped:"," padded string ".strip())
```

```
Capitalized: Hello
Uppercase: WORLD
Right justified:          hello world
Centered:     hello world
Replaced: hello python
Stripped: padded string
```

13. Create and use Python lists. Perform tasks like appending elements, indexing, slic- ing, and iterating through the list.

[24]:
```
my_list=[1,2,3,4]
my_list.append(6)
print("list",my_list)
print("Indexing:",my_list[1])
print("Slicing:",my_list[1:3])
for item in my_list:
    print("List Item:",item)
```

```
list [1, 2, 3, 4, 6]
Indexing: 2
Slicing: [2, 3]
List Item: 1
List Item: 2
List Item: 3
List Item: 4
List Item: 6
```

14. Create and use Python tuples. Perform tasks like indexing, slicing, and concatena- tion

[26]:
```
my_tuple=(1,3,4)
print("Tuple:",my_tuple)
print("Indexing:",my_tuple[0])
print("Slicing :",my_tuple[:2])
tuple_concat=my_tuple + (4,5)
```

```
print("Concated tuple:",tuple_concat)
```

```
Tuple: (1, 3, 4)
Indexing: 1
Slicing : (1, 3)
Concated tuple: (1, 3, 4, 4, 5)
```

15. Create and use Python sets. Perform tasks like accessing, adding, deleting set ele- ments.

[27]:
```
my_set={1,2,3}
my_set.add(4)
my_set.discard(2)
print("Set:",my_set)
for value in my_set:
    print("Set Item:",value)
```

```
Set: {1, 3, 4}
Set Item: 1
Set Item: 3
Set Item: 4
```

16. Create and use Python dictionaries. Perform tasks like adding, updating, and re- moving key-value pairs, and accessing values.

[28]:
```
my_dict = {"name": "Devika", "age": 21}
my_dict["city"] = "Kollam"
my_dict["age"] = 22
del my_dict["name"]

print("Dictionary:", my_dict)
print("Access city:", my_dict.get("city"))
```

```
Dictionary: {'age': 22, 'city': 'Kollam'}
Access city: Kollam
```

17. Define simple functions with parameters and return values.

[29]:
```
def add(x, y):
    return x + y

result = add(3, 4)
print("Addition Result:", result)
```

```
Addition Result: 7
```

18. Call functions with different arguments and use the returned results.

[30]:
```
print("Sum 10 + 20:", add(10, 20))
print("Sum -5 + 7:", add(-5, 7))
```

```
Sum 10 + 20: 30
Sum -5 + 7: 2
```

19. Write functions that accept other functions as arguments.

[31]:
```python
def apply_function(f, x, y):
    return f(x, y)

print("Using apply_function with add:", apply_function(add, 5, 15))
```

```
Using apply_function with add: 20
```

20. Define and use Python classes. Include tasks like creating a class, defining methods, and creating instances

[33]:
```python
class Person:
    def __init__(self, name):
        self.name = name

    def greet(self):
        return f"Hello, my name is {self.name}"

person1 = Person("Amina")
print(person1.greet())
```

```
Hello, my name is Amina
```

21. Implement class inheritance and method overriding.

[34]:
```python
class Student(Person):
    def greet(self):
        return f"Hi, I'm {self.name} and I'm a student."

student1 = Student("Bob")
print(student1.greet())
```

```
Hi, I'm Bob and I'm a student.
```

22. Create a class with class variables and instance variables, and demonstrate their usage.

[35]:
```python
class Demo:
    class_variable = 0  # Class variable

    def __init__(self, value):
        self.instance_variable = value  # Instance variable
        Demo.class_variable += 1

demo1 = Demo(10)
demo2 = Demo(20)
```

```
print("Class Variable:", Demo.class_variable)
print("Instance Variable (demo1):", demo1.instance_variable)
print("Instance Variable (demo2):", demo2.instance_variable)
```

```
Class Variable: 2
Instance Variable (demo1): 10
Instance Variable (demo2): 20
```

[ ]: