Assignment_1

May 11, 2023

[4]:
```
# Q1. Create one variable containing following type of data:
# (i) string
# (ii) list
# (iii) float
# (iv) tuple

# Ans :-

# string
my_string = "Sanket Rathod"

# list
my_list = [9, 5, 2023, "Data", "Science", "Assignment"]

# float
my_float = 10.27

# tuple
my_tuple = (11, 5, 2023, "Pw", "Skills", "Lab")
```

[2]:
```
# Q2. Given are some following variables containing data:
var1 = ''
var2 = '[ DS , ML , Python]'
var3 = [ 'DS' , 'ML' , 'Python' ]
var4 = 1.

# What will be the data type of the above given variable.

#Ans :-

print(type(var1)) # String
print(type(var2)) # String
print(type(var3)) # List
print(type(var4)) # Flot
```

```
<class 'str'>
<class 'str'>
<class 'list'>
```

```
<class 'float'>
```

[1]:
```
# Q3. Explain the use of the following operators using an example:
# (i) /
# (ii) %
# (iii) //
# (iv) **

# Ans :-

# (i) /: The division operator / is used to divide one number by another. It␣
 ↪returns the quotient as a floating-point number. Here's an example:
# division
a = 10
b = 3
c = a / b
print(c)  # output: 3.3333333333333335

#(ii) %: The modulus operator % is used to get the remainder of division. It␣
 ↪returns the remainder after dividing the first number by the second. Here's␣
 ↪an example:
# modulus
a = 10
b = 3
c = a % b
print(c)  # output: 1

#(iii) //: The floor division operator // is used to divide two numbers and␣
 ↪return the quotient as an integer, discarding any remainder. Here's an␣
 ↪example:
# floor division
a = 10
b = 3
c = a // b
print(c)  # output: 3

#(iv) **: The exponentiation operator ** is used to raise a number to a power.␣
 ↪The first operand is the base and the second operand is the exponent. Here's␣
 ↪an example:
# exponentiation
a = 2
b = 3
c = a ** b
print(c)  # output: 8
```

```
3.3333333333333335
1
```

```
3
8
```

[1]:
```
# Q4. Create a list of length 10 of your choice containing multiple types of␣
 ↪data. Using for loop print the element and its data type.
my_list = [1, 2.5, "pw", True, [3, 4], {"name": "Sanket", "age": 21}, None,␣
 ↪"Data scientist", 5, (6, 7)]

for element in my_list:
    print(f"{element} is of type {type(element)}")
```

```
1 is of type <class 'int'>
2.5 is of type <class 'float'>
pw is of type <class 'str'>
True is of type <class 'bool'>
[3, 4] is of type <class 'list'>
{'name': 'Sanket', 'age': 21} is of type <class 'dict'>
None is of type <class 'NoneType'>
Data scientist is of type <class 'str'>
5 is of type <class 'int'>
(6, 7) is of type <class 'tuple'>
```

[1]:
```
# Q5. Using a while loop, verify if the number A is purely divisible by number␣
 ↪B and if so then how many times it can be divisible.
A = 36
B = 4

while A % B == 0:
    C=int(A/B)
    print(f"A is divisible by B, and can be divided {C} times.")
    break
else:
    print("A is not divisible by B.")
```

```
A is divisible by B, and can be divided 9 times.
```

[1]:
```
# Q6. Create a list containing 25 int type data. Using for loop and if-else␣
 ↪condition print if the element is divisible by 3 or not.
numbers = [1, 3, 4, 6, 7, 9, 10, 12, 14, 15, 17, 18, 20, 21, 23, 24, 26, 27,␣
 ↪29, 30, 32, 33, 35, 36, 38]

for num in numbers:
    if num % 3 == 0:
        print(f"{num} is divisible by 3.")
    else:
        print(f"{num} is not divisible by 3.")
```

```
1 is not divisible by 3.
```

```
3 is divisible by 3.
4 is not divisible by 3.
6 is divisible by 3.
7 is not divisible by 3.
9 is divisible by 3.
10 is not divisible by 3.
12 is divisible by 3.
14 is not divisible by 3.
15 is divisible by 3.
17 is not divisible by 3.
18 is divisible by 3.
20 is not divisible by 3.
21 is divisible by 3.
23 is not divisible by 3.
24 is divisible by 3.
26 is not divisible by 3.
27 is divisible by 3.
29 is not divisible by 3.
30 is divisible by 3.
32 is not divisible by 3.
33 is divisible by 3.
35 is not divisible by 3.
36 is divisible by 3.
38 is not divisible by 3.
```

[2]:
```
# Q7. What do you understand about mutable and immutable data types? Give␣
 ↪examples for both showing this property.

# Ans :-
'''In programming, mutable and immutable data types are used to describe␣
 ↪whether a data type can be changed (mutable)
or not (immutable) after it has been created. Immutable data types are those␣
 ↪that cannot be modified after they are created.
Any attempt to modify them results in a new object being created.'''

# Examples of immutable data types include:
'''my_tuple = (1, 2, 3)
my_tuple[0] = 4 # This will raise a TypeError because tuples are immutable'''

'''Mutable data types, on the other hand, can be changed after they are created.
 ↪
Any modification made to the object affects the original object itself.'''

# Examples of mutable data types include:
my_list = [1, 2, 3]
my_list.append(4)  # This adds a new element to the list
print(my_list)  # Output: [1, 2, 3, 4]
```

```
[1, 2, 3, 4]
```