asssesment 1

July 9, 2023

[6]:
```
#A1. Here is an example of creating variables of different data types:
```

[1]:
```
# (i) string
string_variable = "Hello, World!"
```

[2]:
```
# (ii) list
list_variable = [1, 2, 3, 4, 5]
```

[3]:
```
# (iii) float
float_variable = 3.14
```

[4]:
```
# (iv) tuple
tuple_variable = (10, 20, 30)
```

[7]:
```
#A2. The data types of the given variables are:
```

[ ]:
```
# (i) var1
type(var1)  # <class 'str'>
```

[ ]:
```
# (ii) var2
type(var2)  # <class 'str'>
```

[ ]:
```
# (iii) var3
type(var3)  # <class 'list'>
```

[ ]:
```
# (iv) var4
type(var4)  # <class 'float'>
```

[10]:
```
#A3. Operators:
```

[11]:
```
result = 10 / 3
print(result)  # 3.3333333333333335
```

```
3.3333333333333335
```

[12]:
```
#(ii) % is the modulus operator, which returns the remainder of division.
```

[13]:
```
remainder = 10 % 3
print(remainder)  # 1
```

```
1
```

[14]:
```
#(iii) // is the floor division operator, which performs integer division and␣
↪rounds the result down to the nearest whole number.
```

[15]:
```
result = 10 // 3
print(result)  # 3
```

```
3
```

[16]:
```
#(iv) ** is the exponentiation operator, which raises the left operand to the␣
↪power of the right operand.
```

[17]:
```
result = 2 ** 3
print(result)  # 8
```

```
8
```

[18]:
```
#A4. Here is an example of creating a list with multiple types of data and␣
↪printing each element with its data type using a for loop:
```

[19]:
```
my_list = [1, 'two', 3.14, True, [4, 5, 6], {'name': 'John'}]

for element in my_list:
    print(element, type(element))
```

```
1 <class 'int'>
two <class 'str'>
3.14 <class 'float'>
True <class 'bool'>
[4, 5, 6] <class 'list'>
{'name': 'John'} <class 'dict'>
```

[20]:
```
#A5. Here is an example of using a while loop to verify if number A is␣
↪divisible by number B and counting the number of divisions:
```

[21]:
```
A = 16
B = 4
count = 0

while A % B == 0:
    A = A / B
    count += 1

print("A is divisible by B", count, "times.")
```

```
A is divisible by B 2 times.
```

[22]:
```
#A6. Here is an example of creating a list of 25 integers and using a for loop␣
↪with if-else condition to check if each element is divisible by 3:
```

[23]:
```
my_list = list(range(1, 26))

for element in my_list:
    if element % 3 == 0:
        print(element, "is divisible by 3")
    else:
        print(element, "is not divisible by 3")
```

```
1 is not divisible by 3
2 is not divisible by 3
3 is divisible by 3
4 is not divisible by 3
5 is not divisible by 3
6 is divisible by 3
7 is not divisible by 3
8 is not divisible by 3
9 is divisible by 3
10 is not divisible by 3
11 is not divisible by 3
12 is divisible by 3
13 is not divisible by 3
14 is not divisible by 3
15 is divisible by 3
16 is not divisible by 3
17 is not divisible by 3
18 is divisible by 3
19 is not divisible by 3
20 is not divisible by 3
21 is divisible by 3
22 is not divisible by 3
23 is not divisible by 3
24 is divisible by 3
25 is not divisible by 3
```

[24]:
```
#A7. In Python, mutable data types can be modified after they are created,␣
↪while immutable data types cannot be modified once they are created.
```

[25]:
```
#Examples of mutable data types:

#1 List
```

[26]:
```
my_list = [1, 2, 3]
my_list.append(4)
print(my_list)  # [1, 2, 3, 4]
```

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

[27]:
```
#2 Dictionary
```

[28]:
```
my_dict = {'name': 'John', 'age': 25}
my_dict['age'] = 26
print(my_dict)  # {'name': 'John', 'age': 26}
```

```
{'name': 'John', 'age': 26}
```

[29]:
```
#Examples of immutable data types:

#1 String:
```

[30]:
```
my_string = "Hello"
my_string += " World"
print(my_string)  # "Hello World"
```

```
Hello World
```

[31]:
```
#2 Tuple:
```

[ ]:
```
my_tuple = (1, 2, 3)
# Trying to modify a tuple will result in an error
my_tuple[0] = 4  # Raises a TypeError
```

[ ]:
```

```