

Solutions

1.1 Variables and Dynamic Typing

Reference example

```
value = 7
print(type(value).__name__)
value = "seven"
print(type(value).__name__)
```

1.1.1 - Trace

Determine the exact output and explain the state changes.

```
a = [1, 2]
b = a
b.append(3)
print(a)
```

Output

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

Both names refer to the same mutable list.

1.1.2 - Write code

Make an independent copy of [1, 2], append 3 only to the copy, and print both.

A shallow copy is enough for this list of immutable integers.

```
a = [1, 2]
b = a.copy()
b.append(3)
print(a)
print(b)
```

Expected output

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

Solutions

1.2 Numeric Types and Division

Reference example

```
print(7 / 2)
print(7 // 2)
print(-7 // 2)
print(-7 % 2)
```

1.2.1 - Trace

Determine the exact output and explain the state changes.

```
print(11 // 4)
print(11 % 4)
print(2 ** 3)
```

Output

```
2
3
8
```

Floor division gives 2, remainder gives 3, and exponentiation gives 8.

1.2.2 - Write code

Given 135 seconds, print whole minutes and remaining seconds.

Use floor division and remainder.

```
seconds = 135
print(seconds // 60, seconds % 60)
```

Expected output

```
2 15
```

Solutions

1.3 Strings and Formatting

Reference example

```
name = "Ada"
score = 7 / 2
print(f"{name}: {score:.2f}")
```

1.3.1 - Trace

Determine the exact output and explain the state changes.

```
word = "Python"
print(word[1:4])
print(word[-1])
```

Output

```
yth
n
```

The slice includes indices 1 through 3; index -1 is the last character.

1.3.2 - Write code

Format the price 12.5 as Price: \$12.50.

Use two decimal places in an f-string.

```
price = 12.5
print(f"Price: ${price:.2f}")
```

Expected output

Price: \$12.50

Solutions

1.4 Conditionals and Loops

Reference example

```
total = 0
for value in range(1, 6):
    if value % 2 == 0:
        total += value
print(total)
```

1.4.1 - Trace

Determine the exact output and explain the state changes.

```
n = 1
while n < 10:
    n *= 2
print(n)
```

Output

16

The successive values are 2, 4, 8, 16. The next condition check ends the loop.

1.4.2 - Write code

Create and print the list of odd integers from 1 through 7.

Use a stepped range whose stop excludes 8.

```
odds = list(range(1, 8, 2))
print(odds)
```

Expected output

[1, 3, 5, 7]