Exercises 1. What is the value of each of the following expressions?

a. "12" + "3"

b. str(12) + "3"

c. int("12") + 3

d. 3 + float("0.14159")

e. str(2) + ".71828"

f. 3 * "blah"

g. "blah" * 3 != "blahblahblahblah"

Exercises 2. If x = 5, what is the value of x after each of the following statements is executed?

a. x = x ** 3 (alternatively as x = x**3)

b. x = x * 3 (alternatively as x *= 3)

c. x = x / 3 (alternatively as x /= 3)

d. x = x // 3 (alternatively as x //= 3)

e. x = x % 3 (alternatively as x %= 3)

f. x = x + 3 (alternatively as x += 3)

g. x = x - 3 (alternatively as x -= 3)

Exercise 3. Give the value of x after the execution of each of the following sequences:

a.

```python
x = 1
x += x
x += x
x += x
```

b.

```python
x = True
x = not x
x = not x
x = not x
```

c.

```python
x = 2
x *= x
x *= x
x *= x
```

Exercise 4. Write the following numbers using scientific notation in Python:

a. 37,000,000

b. 0.000059
Exercises (Basic Data Types)

Exercise 5. What is the value of each of the following expressions?

a. \(4 + 6 \div 2\)
b. \((4 + 6) \div 2\)
c. \((3 + 6) \cdot 6 - 9 \div 3 + 20\)
d. \(4 + 3 \cdot 2 \div 3\)
e. \((4 + 3) \cdot 2 \div 3\)
f. \(2 ** 1 ** 2\)
g. \((2 ** 1) ** 2\)
h. \(3 < 11 \text{ and } 11 < 8 \text{ or } 8 > 3\)
i. \(-5 \cdot 2 + 34 > 6 + 8 \div 2 \cdot 3\)

Exercise 6. Write a program called reciprocal_division.py that accepts two integers \(x\) and \(y\) as command-line arguments, and writes True to standard output if either number divides the other, and False otherwise.

```
$ python3 reciprocal_division.py 3 4
False
$ python3 reciprocal_division.py 6 3
True
```

Exercise 7. Write a program called distance.py that accepts two floats \(x\) and \(y\) as command-line arguments, and writes to standard output the Euclidean distance of the point \((x, y)\) to the origin \((0, 0)\), computed as \(\sqrt{x^2 + y^2}\).

```
$ python3 distance.py 3 4
5.0
$ python3 distance.py 6 8
10.0
```

Exercise 8. Write a program called sum_of_sines.py that accepts an angle \(t\) (float) in degrees as command-line argument, and writes to standard output the value of \(\sin(2t) + \sin(3t)\).

```
$ python3 sum_of_sines.py 30
1.8660254037844386
$ python3 sum_of_sines.py 60
0.8660254037844386
```

Exercise 9. Write a program called spring.py that accepts \(m\) (int) and \(d\) (int) as command-line arguments, and writes True if day \(d\) of month \(m\) is between 3/20 (inclusive) and 6/20 (inclusive), and False otherwise.

```
$ python3 spring.py 3 19
False
$ python3 spring.py 4 15
True
```

Exercise 10. Write a program called displacement.py that accepts \(x_0\) (float), \(v_0\) (float), and \(t\) (float) as command-line arguments, and writes to standard output the value of \(x_0 + v_0t - gt^2/2\), where \(g\) is the constant 9.80665. This value is the displacement in meters after \(t\) seconds when an object is thrown straight up from initial position \(x_0\) at velocity \(v_0\) meters per second.
Exercises (Basic Data Types)

Exercise 11. Write a program called `compound_interest.py` that accepts \( p \) (float), \( r \) (float), and \( t \) (float) as command-line arguments, and writes to standard output the amount of money you would have after \( t \) years if you invested \( p \) dollars at an annual interest rate \( r \) compounded continuously. The desired value is computed as \( pe^{rt} \).

```
γ_~/workspace/ipp_exercises
$ python3 compound_interest.py 1000 0.04 1
1040.8107741923882
```

Exercise 12. Write a program called `random_gaussian.py` that writes to standard output a random number \( r \) drawn from the Gaussian distribution. One way to do so is to use the Box-Muller formula \( r = \sin(2\pi v)(-2\ln u)^{1/2} \), where \( u \) and \( v \) are real numbers between 0 and 1 generated using the `stdrandom.uniformFloat()` function.

```
γ_~/workspace/ipp_exercises
$ python3 random_gaussian.py
0.0657799651044775
$ python3 random_gaussian.py
0.19601816300230704
```

Exercise 13. Write a program called `order_check.py` that accepts three floats \( x \), \( y \), and \( z \) as command-line arguments, and writes `True` to standard output if the values are in strictly ascending (\( x < y < z \)) or descending (\( x > y > z \)) order, and `False` otherwise.

```
γ_~/workspace/ipp_exercises
$ python3 order_check.py 3 2 1
True
$ python3 order_check.py 1 3 2
False
```

3 / 6
Solutions

Solution 1.

a. "123"

b. "123"

c. 15

d. 3.14159

e. 2.71828

f. "blahblahblah"

g. True

Solution 2.

a. 125

b. 15

(c. 1.6666666666666667

d. 1

e. 2

f. 8

g. 2

Solution 3.

a. 8

b. False

c. 256

Solution 4.

a. 3.7e7

b. 5.9e-5

Solution 5.

a. 7.0

b. 5.0

c. 71.0

d. 6
e. 4
f. 2
g. 4
h. True
i. False

Solution 6.

```python
import sys
import stdio
x = int(sys.argv[1])
y = int(sys.argv[2])
stdio.writeln(y % x == 0 or x % y == 0)
```

Solution 7.

```python
import math
import sys
import stdio
x = float(sys.argv[1])
y = float(sys.argv[2])
stdio.writeln(math.sqrt(x ** 2 + y ** 2))
```

Solution 8.

```python
import math
import sys
import stdio

t = float(sys.argv[1])
stdio.writeln(math.sin(math.radians(2 * t)) + math.sin(math.radians(3 * t)))
```

Solution 9.

```python
import sys
import stdio

m = int(sys.argv[1])
d = int(sys.argv[2])

check = (m == 3 and d >= 20 and d <= 31) or 
(m == 4 and d >= 1 and d <= 30) or 
(m == 5 and d >= 1 and d <= 31) or 
(m == 6 and d >= 1 and d <= 20)

stdio.writeln(check)
```

Solution 10.
Exercises (Basic Data Types)

Solution 11.

```python
import math
import sys
import stdio

p = float(sys.argv[1])
r = float(sys.argv[2])
t = float(sys.argv[3])

stdio.writeln(p * math.exp(r * t))
```

Solution 12.

```python
import math
import stdio
import stdrandom

u = stdrandom.uniformFloat(0, 1)
v = stdrandom.uniformFloat(0, 1)
r = math.sin(math.radians(2 * math.pi * v)) * (-2 * math.log(u)) ** (1 / 2)

stdio.writeln(r)
```

Solution 13.

```python
import sys
import stdio

x = float(sys.argv[1])
y = float(sys.argv[2])
z = float(sys.argv[3])

ascending = x < y and y < z
descending = x > y and y > z

stdio.writeln(ascending or descending)
```