Exception Handling in Python

Standard difficulty · 20 questions · 40 marks · Time: 42 minutes

Name: _____ Date: _____ Score: ____ / 40

Show your working for Sections B and C. Answers are on the last page — try every question first!

Section A — Choose the correct answer — 8 × 1 = 8 marks

1. An error in the rules of writing Python (such as a missing colon) is a:

(a) logical error **(b)** ValueError
(c) exception at run time **(d)** syntax error

2. Which exception does this code raise?

```
x = float('pi')
```

(a) ValueError **(b)** NameError **(c)** IOError **(d)** TypeError

3. What does this print? (Output lines are separated by commas.)

```
L = [40, 10, 88]
try:
    n = int('five')
    print(30 // n)
    print(L[n])
except ZeroDivisionError:
    print('Cannot divide')
except ValueError:
    print('Not a number')
except IndexError:
    print('Bad index')
else:
    print('All fine')
finally:
    print('Done')
```

(a) Not a number, Done **(b)** Not a number
(c) 30, Bad index, Done **(d)** 30, 0, All fine, Done

4. What does this print? (Output lines are separated by commas.)

```
def check(n):
    try:
        return 46 // n
    except ZeroDivisionError:
        return 0
    finally:
        print('checked')

print(check(7))
```

(a) 6, checked **(b)** checked, 0 **(c)** checked, 6 **(d)** 6

5. What does this print? (Output lines are separated by commas.)

```
def setMarks(m):
    if m < 0 or m > 80:
        raise ValueError('Marks out of range')
    return m

try:
    print(setMarks(68))
    print(setMarks(99))
    print(setMarks(49))
except ValueError as e:
    print(e)
```

(a) 68, Marks out of range
(b) 68, Marks out of range, 49
(c) 68, 99, 49
(d) Marks out of range

6. What does this print?

```
data = ['9.5', 'six', '39', 'n/a', '48', '27']
good = 0
bad = 0
for x in data:
    try:
        good = good + int(x)
    except ValueError:
        bad = bad + 1
print(good, bad)
```

(a) 115 3
(b) 114 2
(c) 114 3
(d) 114 4

7. How many times is skip printed?

```
for d in [9, 0, 0, 11]:
    try:
        q = 99 // d
    except ZeroDivisionError:
        print('skip')
    else:
        print(q)
```

(a) 2
(b) 3
(c) 4
(d) 1

8. Which error does Python report before running this code?

```
print('8 apples)
```

(a) TypeError
(b) NameError
(c) SyntaxError
(d) ValueError

Section B — Short answer — 8 × 2 = 16 marks

9. What is the difference between a syntax error and an exception?

10. Name the exception raised by this code.

```
L = [1, 2, 5]
print(L[6])
```

11. What does this print? Write the lines in order, separated by commas.

```
L = [76, 72, 56]
try:
    n = int('8')
    print(45 // n)
    print(L[n])
except ZeroDivisionError:
    print('Cannot divide')
except ValueError:
    print('Not a number')
except IndexError:
    print('Bad index')
else:
    print('All fine')
finally:
    print('Done')
```

12. What does this print? Give (a) the first line (b) the second.

```
def check(n):
    try:
        return 82 // n
    except ZeroDivisionError:
        return 0
    finally:
        print('checked')

print(check(9))
```

13. What does this print? Give (a) the value of good (b) the value of bad.

```
data = ['70', '', 'six', '4', '37', '29']
good = 0
bad = 0
for x in data:
    try:
        good = good + int(x)
    except ValueError:
        bad = bad + 1
print(good, bad)
```

14. Write the four lines this prints, separated by commas.

```
for d in [9, 3, 0, 0]:
    try:
        q = 74 // d
    except ZeroDivisionError:
        print('skip')
    else:
        print(q)
```

15. Write the lines this prints, in order, separated by commas.

```
def setMarks(m):
    if m < 0 or m > 50:
        raise ValueError('Marks out of range')
    return m

try:
    print(setMarks(36))
    print(setMarks(51))
    print(setMarks(15))
except ValueError as e:
    print(e)
```

16. Write the letters this prints, in order, separated by commas.

```
try:
    x = 2
    y = x / 0
    print('A')
except ZeroDivisionError:
    print('B')
else:
    print('E')
finally:
    print('C')
print('D')
```

Section C — Word problems — 4 × 4 = 16 marks

17. Explain the try … except … else … finally statement with a program that divides two numbers typed by the user.

18. Write the output of each program, one line under another.

(a)

```
L = [79, 20, 88]
try:
    n = int('0')
    print(45 // n)
    print(L[n])
except ZeroDivisionError:
    print('Cannot divide')
except ValueError:
    print('Not a number')
except IndexError:
    print('Bad index')
else:
    print('All fine')
finally:
    print('Done')
```

(b)

```
for d in [2, 0, 0, 6]:
    try:
        q = 35 // d
    except ZeroDivisionError:
        print('skip')
    else:
        print(q)
```

19. (a) What does the first program print? (b) What does the second print? (c) When does a finally block run?

(i)

```
def check(n):
    try:
        return 94 // n
    except ZeroDivisionError:
        return 0
    finally:
        print('checked')

print(check(9))
```

(ii)

```
def setMarks(m):
    if m < 0 or m > 100:
        raise ValueError('Marks out of range')
    return m

try:
    print(setMarks(27))
    print(setMarks(116))
    print(setMarks(85))
except ValueError as e:
    print(e)
```

20. (a) What does this program print? (b) Which items of data raise the exception? (c) What would happen if the try and except lines were removed?

```
data = ['', 'ab', '34', '5.5', '40']
good = 0
bad = 0
for x in data:
    try:
        good = good + int(x)
    except ValueError:
        bad = bad + 1
print(good, bad)
```

Want another sheet on this chapter? In the Ambiful Worksheets app every sheet is new, with its answer key, and your child never gets the same question twice.
Scan the code to get the free app, or open **worksheets.ambiful.com/get**

Answer key

Exception Handling in Python · AW-12CS-EXH-2026-V1 · For parents — fold back or tear off before handing the sheet to your child.

Q	Answer	Working
1	**(d) syntax error**	—
2	**(a) ValueError**	the string cannot be read as a number
3	**(a) Not a number, Done**	int() of a word raises ValueError
4	**(c) checked, 6**	finally runs before the function hands back its value
5	**(a) 68, Marks out of range**	once raise runs, the rest of the try block is skipped and except prints the message
6	**(c) 114 3**	int() raises ValueError for a string with a decimal point, a word or an empty string
7	**(a) 2**	only d = 0 raises ZeroDivisionError
8	**(c) SyntaxError**	the string is never closed
9	**A syntax error breaks the rules of the language and stops the program from running at all; an exception happens while a correct program runs, e.g. dividing by zero.**	—
10	**IndexError**	L has indices 0 to 2 only
11	**5, Bad index, Done**	L has only indices 0 to 2, so L[8] raises IndexError after the division prints
12	**(a) checked; (b) 9**	finally runs before the function hands back its value
13	**(a) 140; (b) 2**	int() accepts only whole-number strings: one with a decimal point, a word or '' raises ValueError
14	**8, 24, skip, skip**	else runs only when the division worked
15	**36, Marks out of range**	raise jumps straight to the matching except
16	**B, C, D**	x / 0 raises ZeroDivisionError, so A and else are skipped; finally always runs; then the program carries on
17	**try: a = int(input()); b = int(input()); c = a / b; except ZeroDivisionError: print('b cannot be 0'); except ValueError: print('Enter whole numbers'); else: print(c); finally: print('Done'). Only one except runs; else runs when nothing went wrong; finally always runs.**	—
18	**(a) Cannot divide, Done; (b) 17, skip, skip, 5**	n is 0, so // raises ZeroDivisionError
19	**(a) checked, 10; (b) 27, Marks out of range; (c) always: after the try (and any except or else) part, whether or**	finally runs on every path

Q	Answer	Working
	not an exception was raised, even when try returns.	
20	**(a) 74 3; (b) '', 'ab', '5.5'; (c) the first of them would raise an unhandled ValueError, the program would stop with a traceback and nothing would be printed.**	int() accepts only strings of digits; one with a decimal point, a word or '' raises ValueError