Coffee Break Python

Mastery

>>> Workout

#1 Bestselling Book Series LeanPub 2019

99 Tricky Python Puzzles to Push You to Programming Mastery

Rieger, Chan, Mayer

Coffee Break Python - Mastery Workout

99 Tricky Python Puzzles to Push You to Programming Mastery

Lukas Rieger, Adrian Chan, and Christian Mayer

May 18, 2020

A puzzle a day to learn, code, and play.

Contents

— 1 —

Introduction

The main driver for mastery is neither a character trait nor talent. Mastery comes from intense, structured training. Mark Zuckerberg, the founder of Facebook, reached mastery at a young age as a result of coding for more than 10,000 hours—laying the foundation to a billion-user company at the age of 20. He was committed and passionate about coding and worked long nights to develop his skills. He was anything but an overnight success.

Here's what we, the authors, know about you: you're an aspiring coder, and you seek ways to improve your coding skills. How do we know? Simple statistics: only a small fraction of people reads books in their fields. You don't spend money and time on a book if you don't seek to advance your skills.

Advancing your coding skills is, ultimately, one of the most powerful things you can do to advance the efficiency of society overall and, thus, your financial and social reward from society. Machines work diligently 24 hours per day, seven days per week. If you're a good programmer, your programs will provide a never-ending stream of value to you, your family, your state, your country, and your fellow men. Why? Because machines replicate and automate your ability to create value. Learning to code will give you a powerful tool to make you a respectable member of society and to provide you with confidence and financial freedom.

The *Coffee Break Python - Mastery Workout* is the second workbook and the fifth book of the *Coffee Break Python* series of Python textbooks. In a way, it's an extension of its predecessor *Coffee Break Python Workbook*[1]—but with 99 brand-new code puzzles teaching new Python concepts, it stands on its own.

This book aims to be a stepping stone on your path to becoming a Python master. It contains 15-25 hours of Python training using one of the most efficient learning techniques: *practice testing*. This technique is guaranteed to improve your ability to read, write, and understand Python source code.

The idea is that you solve code puzzles. A *code puzzle*

[1] `https://blog.finxter.com/coffee-break-python-workbook/`

is an educational snippet of source code that teaches a single computer science concept by activating your curiosity and involving you in the learning process. The code puzzles in this book have an intermediate to advanced difficulty level.

How does it work? In essence, you play the Python interpreter and compute the output of each code snippet in your head. Then you check whether you were right. Using the accompanying feedback and explanations, you will adapt and improve your coding skills over time.

To make this idea a reality, we developed the online coding academy `Finxter.com`. The next section explains the advantages of the Finxter method of puzzle-based learning. If you already know about the benefits of puzzle-based learning from previous books and want to dive right into the puzzles, you can skip the following section and start at Chapter 2.

1.1 Why Code Puzzles?

Robust scientific evidence shows that active learning doubles students' learning performance. In a study on this matter, test scores of active learners improved by more than a grade compared to their passive learning

counterparts.[2] Not using active learning techniques wastes your time and hinders you if you want to reach your full potential. Switching to active learning is a simple tweak that instantly improves your performance.

Active learning requires the student to interact with the material, rather than simply consume it. It is student-centric rather than teacher-centric. Great active learning techniques are asking and answering questions, self-testing, teaching, and summarizing.

A popular study shows that one of the best learning techniques is *practice testing*.[3] In this technique, you test your knowledge before you have learned everything. Rather than *learning by doing*, it's *learning by testing*.

The study argues that students must feel safe during these tests. Therefore, the tests must be low-stakes, i.e., students have little to lose. After the test, students get feedback on how well they did. The study shows that practice testing boosts long-term retention. Solving a daily code puzzle is not just another learning technique—it is one of the best.

Although active learning is twice as effective, most books focus on passive learning. The author delivers informa-

[2] `https://en.wikipedia.org/wiki/Active_learning#Research_evidence`

[3] `http://journals.sagepub.com/doi/abs/10.1177/1529100612453266`

tion; the student passively consumes it. Some programming books include active learning elements by adding tests or by asking the reader to try out the code examples. Yet, we've always found this impractical when reading on the train, bus or in bed. If these active elements drop out, learning becomes 100% passive again.

Fixing this mismatch between research and everyday practice drove me to write this book series about puzzle-based learning. In contrast to other books, this book makes active learning a first-class citizen. Solving code puzzles is an inherent active learning technique. You must figure out the solution yourself for every single puzzle. Before you study the correct solution, your knowledge gap is already wide open. Thus, you are mentally prepared to digest new material.

1.2 The Elo Rating for Python

Pick any sport you've always loved to play. How good are you compared to others? The Elo rating—initially developed to rate the skills of chess players—answers this question with surprising accuracy. The higher the Elo rating, the better the chess player. Now, we simply take this concept and use it to rate your Python skills.

Table 1.1 shows the ranks for each Elo rating level. The table is an opportunity for you to estimate your Python

Elo rating	**Rank**
2500	World Class
2400-2500	Grandmaster
2300-2400	International Master
2200-2300	Master
2100-2200	National Master
2000-2100	Master Candidate
1900-2000	Authority
1800-1900	Expert
1700-1800	Professional
1600-1700	Experienced Intermediate
1500-1600	Intermediate
1400-1500	Experienced Learner
1300-1400	Learner
1200-1300	Scholar
1100-1200	Autodidact
1000-1100	Beginner
0-1000	Basic Knowledge

Table 1.1: Elo ratings and skill levels.

skill level. In the following, we'll describe how you can use this book to test your Python skills.

1.3 How to Use This Book

This book contains 99 (tricky) code puzzles and explanations to test and train your Python skills. The Elo ranges from 1500 to 2300 points (between *intermediate* and *Master* level in the table). This book is perfect for you if you are beyond intermediate level. Even experts will improve their speed of code understanding if they follow the outlined strategy.

1.4 How to Test and Train Your Skills?

We recommend you solve at least one code puzzle every day, e.g., as you drink your morning coffee. Then spend the rest of your learning time on real projects that matter to you. The puzzles guarantee that your skills will improve over time, and the real project brings you results.

To test your Python skills, do the following:

1. Track your Elo rating as you read the book and solve the code puzzles. Write your current Elo rating in the book. Start with an initial rating of 1500 if you are an intermediate, and 2000 if you are an advanced Python programmer. If you already

have an online rating on `Finxter.com`, start with that.

2. For each puzzle, if your solution is correct, add the Elo points given with the puzzle. Otherwise, subtract the points from your current Elo number.

3. Figure 1.4 shows how your Elo will change while working through the book. Two factors impact the final rating: how you select your initial rating and how good you perform (the latter being more important).

If you're an intermediate-level coder, solve the puzzles sequentially because they build upon each other. If you don't know about particular Python features, check out our five cheat sheets in the appendix Chapter 14. Alternatively, you can download all five cheat sheets as concise PDFs here: `https://blog.finxter.com/python-cheat-sheets`
Advanced readers can solve the puzzles in the sequence they wish—the Elo rating will work just as well.

Use the following training plan to develop a healthy learning habit with puzzle-based learning.

1. Choose or create a daily trigger, after which you'll solve code puzzles for 10 minutes. Solve code puzzles as you brush your teeth or sit on the train to work, university, or school.

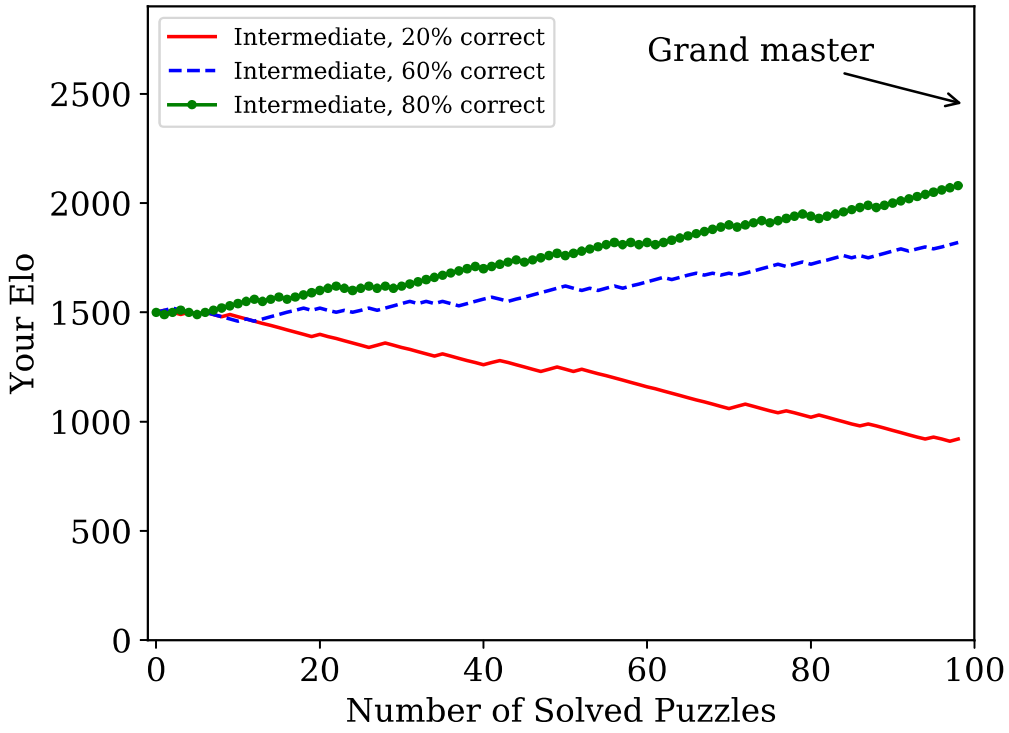


Figure 1.1: Example of how your Elo rating could change while working through the 99 puzzles. Your final Elo rating depends on the percentage of correctly solved puzzles.

2. Scan over the puzzle and ask yourself: what is the unique idea of this puzzle?

3. Dive deeply into the code. Try to understand the purpose of each symbol, even if it seems trivial at first. Avoid being shallow and lazy. Instead, solve each puzzle thoroughly and take your time. It may seem counter-intuitive at first, but to learn faster, you must take your time and allow yourself to dig deep. There is no shortcut.

4. Stay objective when evaluating your solution—we all tend to lie to ourselves.

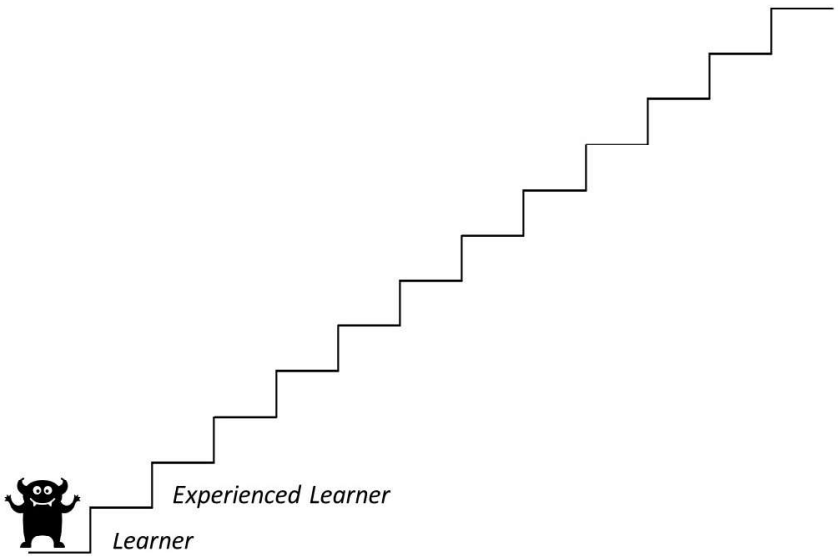
5. Look up the solution and read the explanation with care. Do you understand every aspect of the code? Write questions down and look up the answers later. Or send them to us (`admin@finxter.com`). We will do everything we can to come up with a good explanation.

6. If your solution was 100% correct—including whitespaces, data types, and formatting of the output—you get the Elo points for this puzzle. Otherwise, your solution was wrong, and you must subtract Elo points. This rule is strict because code is either right or wrong. If you miss some whitespace in the wild, you may get an error.

As you follow this simple training plan, your ability to understand source code quickly will improve. You do not have to invest much time because the training plan requires only 10–20 minutes per day. But you must be persistent with your effort. If you get off track, get right back on the next day. When you run out of code puzzles, feel free to checkout `Finxter.com`. The Finxter app has more than 500 hand-crafted code puzzles—and we regularly publish new ones.

— 2 —

Python Elo 1300-1400: *Learner* to *Experienced Learner*



2.1 Basic For Loop

```
# Elo 1348

n, count = 5, 0
for _ in range(n):
    n -= 1
    count += 1
print(count)
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

First, you initialize the variables `n = 5` and `count = 0`. The range of the for loop is from `0` to `4` (inclusive) which makes 5 iterations. Since the variable `count` is incremented by `1` in each iteration, the final value of the variable `count` is `5`.

2.2 Variable Scope III

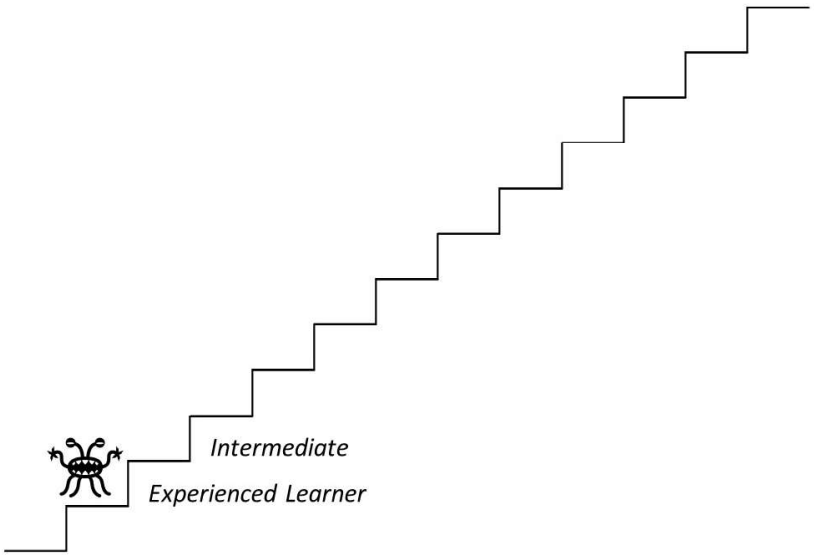
```
# Elo 1367

def swap(a, b):
    return b, a

a, b = 1, 2
a, b = swap(a, b)
print(a - b)
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

The function `swap()` returns a tuple of the swapped values. You assign this tuple back to the variables `a` and `b`, so global variable `a` takes on the value of `b` while global variable `b` takes on the value of `a`. Thus, the output is `a - b = 2 - 1 = 1`

— 3 —

Python Elo 1400-1500: *Experienced Learner* to *Intermediate*



3.1 Boolean Conversion

```
# Elo 1487

x = 0
if False or [False] or (False):
    x += 1
print(x)
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

Any non-empty iterable has a Boolean value of `True`, therefore the code inside the if statement is executed which increments the value of variable `x` by `1`. This produces the final output `1`.

3.2 Tuple Constructor

```
# Elo 1489

s = 'abc'
for _ in range(10):
    s = tuple(s)
print(len(s))
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

First, you create string `'abc'` and store it in the variable `s`. Second, you create a new tuple using `s` in the for loop. The tuple constructor uses the characters of the string to create the tuple `('a', 'b', 'c')`. You repeat this exact procedure nine times—without really changing the tuple. So, the function `len()` returns `3`.

3.3 Popping List Elements

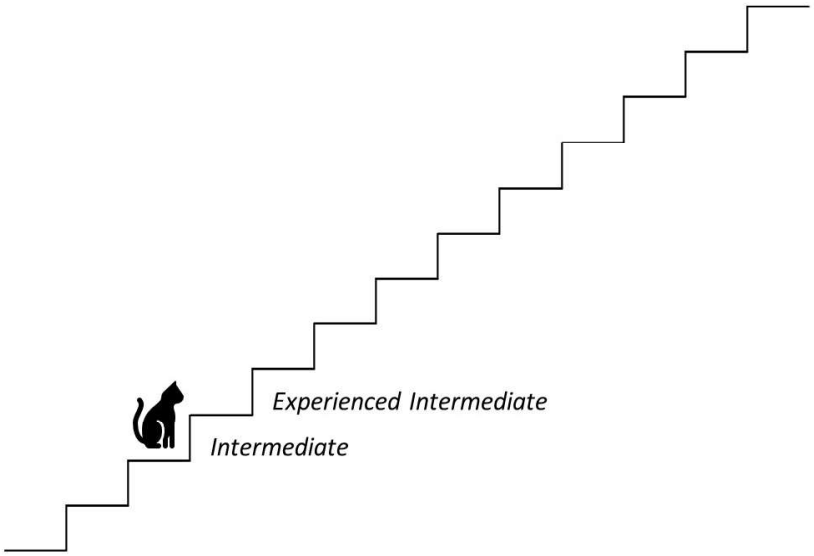
```
# Elo 1499

t = [1, 2, 3, 4]
x = t.pop() > t.pop() > t.pop()
t.append(x)
print(len(t))
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

Each call of the function `pop()` on a list returns and removes its last element. From left to right, the first call returns `4`, the second `3`, and the third `2`. You store the result of the comparison in the variable `x` and append it to list `t`. You initialized the list four elements, removed three, and finally appended one. So, the length of list `t` is `2`.

— 4 —

Python Elo 1500-1600: *Intermediate* to *Experienced Intermediate*



4.1 Extending Lists

```
# Elo 1517

t = [1, 2, 3, 4]
x = (t.pop() < t.pop()), t.pop()
t.extend(x)
print(len(t))
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

Each call of the function `pop()` returns and removes the last element from the list. From left to right, the first call returns `4`, the second `3`, and the third `2`. Variable `x` holds a tuple containing the result of the comparison `4 < 3` and `2`. The method `extend()` adds both tuple elements to list `t`. List `t` starts with four elements. You remove three of them. Then, you add both tuple elements. In the end, list `t` contains three elements, so the output is `3`.

4.2 list.extend() vs. list.append()

```
# Elo 1539

t = [0, 1, 2]
t.extend([])
t.append([])
print(len(t))
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

The trick of this puzzle is to understand the difference between the two list methods `extend()` and `append()`.

- The method `extend(iter)` takes an iterable `iter` and adds its items to the end of the list.
- The method `append(element)` takes one argument `element` and adds this one element to the end of the list. So if you append a list to another list, the list would be appended as a single element—creating a list of lists.

In the puzzle, you call the method `extend()` on list `t` with an empty list which doesn't change the list. Then, you call `append()` with an empty list which adds the empty list as an element at the end of `t`. Thus, the length of the list `t` is `4`—three integers 0, 1, and 2, plus the empty list `[]`.

4.3 List Objects

```
# Elo 1545

t = [0, 1, [2], 3]
y = t[2]
y.append(20)
print(t[2])
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

The list `t` contains integer values and another list with one element `2`. The expression `t[2]` retrieves the inner list object and stores a reference to it in variable `y`. Because it's a reference, the method `append()` appends to the inner list of list `t`. Thus, when we retrieve the inner list again by calling `t[2]`, it has changed and the output is `[2, 20]`.

4.4 Python Basic Set Operations

```
# Elo 1587

a = [1, 1, 1, 2, 3, 3]
b = [0, 0, 0, 1, 3, 3]
print(set(a) >= set(b))
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

Converting list `a` into a set yields the result `{1, 2, 3}` and for list `b` it yields the result `{0, 1, 3}`. The operator `>=` applied on sets checks if the set on the left side is a superset of the set on the right side. In other words: are all elements in the right set also in the left set? Since this is not the case in this puzzle, the output is `False`.

4.5 Length Arithmetik

```
# Elo 1587

s = '-5\\3'
print(len(s))
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

The escape character
invalidates characters with special meaning. In this puzzle, you use the escape character to invalidate the escape character. If you print the string `s` to the shell, the output will be `'-5
3'`. For the length of a string, the escape character doesn't count. Therefore, you get the output `4`.

4.6 Print Arguments

```
# Elo 1589

t = 'abc'
d = {'sep': '-', 'end': '?'}
print(*t, **d)
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

The function `print()` takes any number of objects to be printed and also some optional arguments. You can use the optional argument `sep` to print a string between each pair of objects. The optional argument `end` can be used to modify the last character, which is printed after all the objects. By default it is the new line character (`\n`).

In the function call `print()`, you unpack the string with the asterisk operator (`*`). Therefore, each character in the string becomes a separate object to be printed. With the double-asterisk operator (`**`), you unpack the dictionary and pass its entries as keyword arguments. The output is, therefore, `a-b-c?`.

4.7 Dictionary Keys

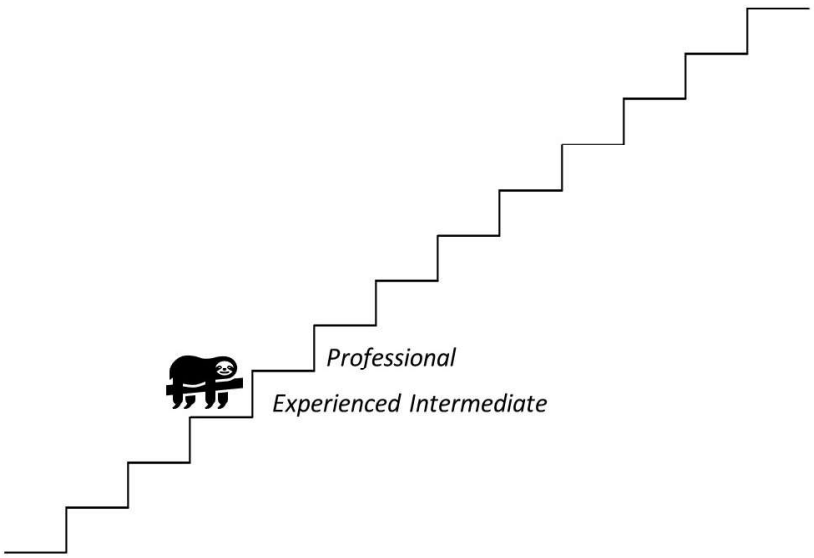
```
# Elo 1598

x = 10
d = {x:0, 11:1}
x = 11
print(d[x])
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

You use the value of variable `x` as a key when creating dictionary `d`. Since the value of variable `x` is a primitive datatype, there are no references to the same integer object, only copies—and changing the value of `x` will not change the key in dictionary `d`. So, when you retrieve the value for key `11`, you get the output `1`.

— 5 —

Python Elo 1600-1700: *Experienced Intermediate* to *Professional*



5.1 Boolean Data Structure and string.find()

```
# Elo 1602

s = 'bed and breakfast'
r = s.find('bed') == False
print(r)
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

In the first line of this puzzle, you initialize variable `s` with the string `'bed and breakfast'`. In the second line, you use the string method `find()` to get the index of the string `'bed'` inside the string `s`. The string `find()` method returns the start index of the searched substring, or `-1` if it doesn't exist. In the string `'bed and breakfast'` the substring `'bed'` starts at index 0, therefore the `find()` method returns `0`.

Next, you compare `0` to `False`—because every integer value can be mapped to a Boolean value: `0` maps to `False`, any other number maps to `True`. Thus, comparing `0 == False` results in `True` which is the output of this puzzle.

5.2 Slicing

```
# Elo 1639

s = 'smart'
print(s[:-100:-2])
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

After initializing variable `s` with the value `'smart'`, you slice it. The slice starts at the beginning of the string and ends at index `-100`. The step size is `-2`, so you take every second element starting with the last element. Python ignores that there is no element with index `-100` and simply takes the whole string `'smart'` from which it extracts every second character starting with the last one. The final output is `'tas'`.

5.3 List Arithmetic I

```
# Elo 1665

tup = 0, 1, 2
lst = [tup]
lst.append(3)
lst.pop(0)
print(len(lst))
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

You create a list `lst` which contains one element: the tuple `tup`. Then, you append an integer element 3 to `lst`. The method `pop(index)` removes and returns the element at position `index`. After calling the method `pop(0)` on the first list element, the list looks like this: `[3]`. As the list `lst` only contains one element, it has a length of `1`.

5.4 List Arithmetic II

```
# Elo 1666

tup = 0, 1, 2
lst = list(tup)
lst.append(3)
lst.pop(0)
print(len(lst))
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

In this puzzle, you use the list class's constructor to instantiate a new list from tuple `tup`. The new list contains each tuple element as a single list element. So the list `lst` looks like this: `[0, 1, 2]`. After appending `3` to the end and removing the first element, the list still has three elements. Therefore, the output is `3`.

5.5 Variable Scope II

```
# Elo 1671

def swap():
    b, a = a, b

a, b = 1, 2
swap()
print(a - b)
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

The official documentation states: *when you make an assignment to a variable in a scope, that variable becomes local to that scope and shadows any similarly named variable in the outer scope.*[1]

When entering the function `swap()`, Python first checks the local scope for the variable names `a` and `b`. As you assign to both variables, it considers both to be in the local scope. But on the right-hand side of the equation, you access those local variables that haven't yet been defined. Consequently, this leads to an error.

To make the code switch the values of the global variables `a` and `b`, you have to tell the function to use the global variables. You can accomplish this by adding inside the function body: `global a; global b`. Without this fix, the code throws an error.

5.6 String Find

```
# Elo 1689

s = 'apple'
print(s.find('p') + s.find('f') + s.find(''))
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

[1] `https://docs.python.org/3/faq/programming.html`

The method `string.find(substring)` returns the index of `substring` in `string` or `-1` if `string` doesn't contain the `substring`. In the puzzle, the first use of the function `find('p')` returns `1` which is the index of the first occurrence of `'p'` in `'apple'`. The second call `find('f')` returns `-1` because the string `'apple'` doesn't contain `'f'`. Finally, for the empty character `''`, you get back index `0`. Summing up these values, you get the output `1 - 1 + 0 = 0`.

5.7 Zip Heterogeneous Lists

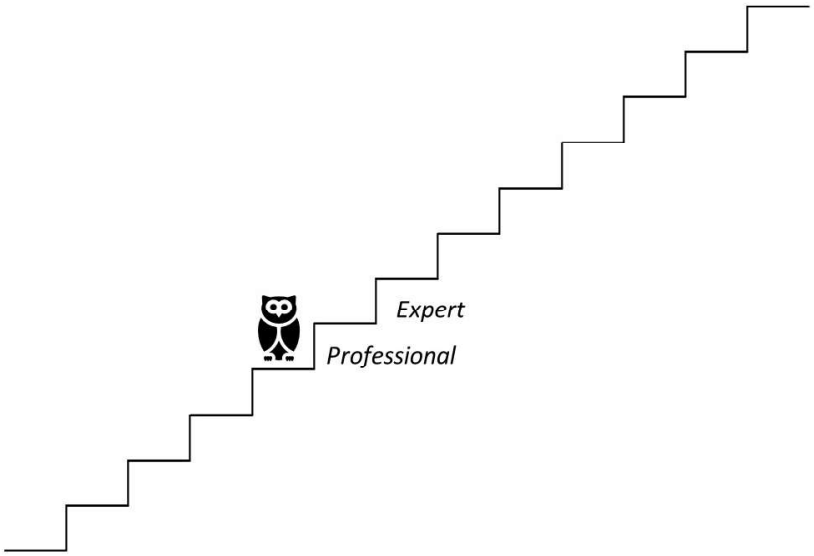
```
# Elo 1699

p = ['abc']
q = ['a', 'b', 'c']
zipped = list(zip(p, q))
print(len(zipped))
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

Python's built-in function `zip()` takes one or more iterables, say `p` and `q`, and returns a list of tuples—by pairing the i-the elements of the iterables. The first entry of the first tuple is the first item of `p`, and the second entry is the first item of `q`. The shorter one of both iterables determines the total length of the zipped result. In the puzzle, list `p` hast one element, whereas list `q` has three elements. Therefore the list `zipped` looks like this: `[('abc', 'a')]` and has a length of `1`.

— 6 —

Python Elo 1700-1800: *Professional* to *Expert*



6.1 Boolean Integer Relationship

```
# Elo 1721

num = 0
for i in range(5, 0, -1):
    num += i > num
print(num)
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

After assigning value `0` to the variable `num`, you create a for-loop using the `range()` function. The range starts at `5` and goes down to `1` because of the negative step size. Inside the loop, you add the result of `i > num` to the current value of `num`. The result of the comparison can either be `True` of `False`. Python converts these values implicitly to `0` (`False`) and `1` (`True`). In the first three iterations the value of variable `i` is smaller than the values in `num`. Therefore, the comparison result is `True` and the value in `num` increases by `1`. As soon as the value in variable `num` reaches `3`, the result of the comparison becomes `False` and you add `0`. The final output is, therefore, `3`.

6.2 Slice Assignment Basics

```
# Elo 1749

t = [1, 2, 3]
t[-1:] = [4]
t[-1] = [5]
print(len(t))
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

First, you use slice assignment to overwrite the final list element with value `4`. Second, you again overwrite the last list element with the list `[5]`. The length of the list never changes since you only exchange elements. Thus, the output is `3`.

6.3 List Multiplication

```
# Elo 1755

t = [0, 1, 2]
t2 = t * 1
t[0] = 100
print(t2)
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

The variable `t` points to the list `[0, 1, 2]` in memory. By multiplying `t` by `1` we create a new list, so that variable `t2` points to another address in memory—a copy of list `t`. This is why the assignment `t[0] = 100` doesn't change the list object to which `t2` points. Thus, you obtain the output `[0, 1, 2]`.

6.4 Integers in Memory

```
# Elo 1766

x = 10
d = {'Peter': x, 'Tom': x, 'Mary': x}
x = 11
print(d['Peter'])
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

You create a dictionary and store it in variable `d`. The dictionary keys are strings, and the dictionary values are copies of variable `x`. Since these values are integers, we don't have references to variable `x` in the dictionary. Thus, changing the value of variable `x` doesn't affect the values in the dictionary. The output is `10`.

6.5 Slice Assignment

```
# Elo 1767

t = [1, 2, 3, 4]
t[2::-2] = [10, 30]
print(t[0])
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

The code modifies a list with four elements using slice assignment. You replace the selected elements with `t[2::-2]` with the values 10 and 30. The first selected element is at starting position 2. As you have negative step size -2, you go from starting position 2 to the next position 2 - 2 = 0. In summary, you replace the element at position 2 with 10, and the element at position 0 with 30—the latter being the output of the puzzle.

6.6 Maximum With Key

```
# Elo 1771

t = [-2, 9, 0, -8]
key = lambda x: -abs(x)
print(max(t, key=key))
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

In this puzzle, you use the optional argument `key` of the `max()` function and pass a function to it named `key`. The function takes one argument `x` and returns the negated absolute value of `x`. After applying the function key to list `t`, you get the values `[-2, -9, 0, -8]` as key values. Thus, the maximum is `0`.

6.7 Maximum With Key

```
# Elo 1777

t = [0, 1, 2, 3]
print(max(t, key=lambda x: x\%2))
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

The built-in function `max()` takes an optional argument `key` that expects a function. This key function computes a value for each element in the input list. The function `max()` operates on the values computed by the key function and returns the maximum depending on those values. In this puzzle, you compute the modulo 2 value for each element in the list, so that you get `1` for odd values and `0` for even values. Since `1` is the first element in the list `t` with maximum key value, the output is `1`.

6.8 Variable Scoping Lists

```
# Elo 1777

t = [0]

def f():
    t.append(10)

t = [1]
f()
print(t)
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

In this puzzle, the function `f` operates on the global variable `t`, so the call `t.append(10)` modifies the list stored in `t`. Since you reassign `t` to `[1]` and then call function `f`, you get the final output `[1, 10]`.

6.9 Dictionary Comprehension

```
# Elo 1787

a, b, c = dict((i, i*2) for i in range(3))
print(a)
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points

You can initialize a dictionary in Python with an iterable of tuples—this is called *dictionary comprehension*. The first tuple entries are the keys and the second tuple values are the values of the dictionary. You assign the three dictionary keys to the three variables `a`, `b`, and `c`. Since the function `range(x)` creates a range from `0` to `x-1` the first key of the dictionary is `0`. Thus, printing variable `a` leads to the output `0`.

6.10 Minimum with Key

```
# Elo 1787

strings = ['flies', 'zzz', 'mammal']
first = min(strings, key=lambda x: len(set(x)))
print(first)
```

What's the output of this code snippet?
Correct: +10 Elo points / Wrong: -10 Elo points