# AI – Python lab 1

AI – Fall, 2023

Your first Python lab is to solve the first eight problems under each of the level 1 tasks in the Python tab at CodingBat.com. That means tabs Warmup-1, String-1, List-1, Logic-1, and all the problems in the top two lines (as the lines have three problems each) and the first two problems on the third line. The complete list of problems appears below.

This lab is assigned on the first day of class. On that day, we will go over the Python constructs that one might use to solve the problems. You do not need any import statements for this lab. On the second day of class, the method for turning in the assignments is covered. Until then, **you should have a single file, and each time you solve a problem, add the complete function definition to the file**. You should not have any semicolons, tabs, imports, in your file, nor "):" except at the end of a line.

When doing this assignment, if you have the luxury of doing it in one sitting, please record the length of time it takes you to do it, as your instructor would like to get a measure of the average time.

It is possible to solve this assignment completely using a subset of the following Python constructs, covered in class the first day:

| | | | | | |
|---|---|---|---|---|---|
| | `=` (assignment) | `(`grouping`)` | `+=` (in place add) | `*=` (in place multiply) | |
| mathy: | `/` (divide) | `*` (multiply) | `%` (modulo) | `+` (add / concatenate) | |
| | `//` (int divide) | `**` (power) | `abs(…)` | | |
| | `\|` (bitwise `or`, and set union) | | `&` (bitwise `and`, and set intersection) | | |
| | `-` (subtract, and set difference) | | `^` (bitwise `xor`, and disjoint set union) | | |
| logical: | `and` | `or` | `not` | | |
| comparisons: | `==` | `<` (less than) | `>` (more than) | `<=` | `>=` |
| set, list: | `set(…)` | `{…}` | `list(…)` | `[…]` | |
| ranges, slice: | `range(…)` | `:` (slice) | | | |
| flow-control: | `if` | `elif` | `else` | `return` | |
| strings: | `"…"` | `'…'` | `startswith` | `endswith` | `format` |
| info: | `len` | `in` | `sum` | `max` | `min` |

Specific problems:

| Warmup-1 | | |
|---|---|---|
| `sleep_in` | `monkey_trouble` | `sum_double` |
| `diff21` | `parrot_trouble` | `makes10` |
| `near_hundred` | `pos_neg` | |

| String-1 | | |
|---|---|---|
| `hello_name` | `make_abba` | `make_tags` |
| `make_out_word` | `extra_end` | `first_two` |
| `first_half` | `without_end` | |

| List-1 | | |
|---|---|---|
| `first_last6` | `same_first_last` | `make_pi` |
| `common_end` | `sum3` | `rotate_left3` |
| `reverse3` | `max_end3` | |

| Logic-1 | | |
|---|---|---|
| `cigar_party` | `date_fashion` | `squirrel_play` |
| `caught_speeding` | `sorta_sum` | `alarm_clock` |
| `love6` | `in1to10` | |

The specific functions are listed on the next two pages. In a few cases, indicated in gray, the problem has been slightly revised from the version on codingbat.com.

Warmup 1:

| Function form | function statement |
|---|---|
| | |
| `sleep_in(weekday, vacation)` | The parameter weekday is True if it is a weekday, and the parameter vacation is True if we are on vacation. We sleep in if it is not a weekday or we're on vacation. Return True if we sleep in. |
| `monkey_trouble(a_smile, b_smile)` | We have two monkeys, a and b, and the parameters a_smile and b_smile indicate if each is smiling. We are in trouble if they are both smiling or if neither of them is smiling. Return True if we are in trouble. |
| `sum_double(a, b)` | Given two int values, return their sum. Unless the two values are the same, then return double their sum. |
| `diff21(n)` | Given an int n, return the absolute difference between n and 21, except return double the absolute difference if n is over 21. |
| `parrot_trouble(talking, hour)` | We have a loud talking parrot. The "hour" parameter is the current hour time in the range 0..23. We are in trouble if the parrot is talking and the hour is before 7 or after 20. Return True if we are in trouble. |
| `makes10(a, b)` | Given 2 ints, a and b, return True if one if them is 10 or if their sum is 10. |
| `near_hundred(n)` | Given an int n, return True if it is within 10 of 100 or 200. Note: abs(num) computes the absolute value of a number. |
| `pos_neg(a, b, negative)` | Given 2 int values, return True if one is negative and one is positive. Except if the parameter "negative" is True, then return True only if both are negative. |

String 1:

| | |
|---|---|
| `hello_name(name)` | Given a string name, e.g. "Bob", return a greeting of the form "Hello Bob!". |
| `make_abba(a, b)` | Given two strings, a and b, return the result of putting them together in the order abba, e.g. "Hi" and "Bye" returns "HiByeByeHi". |
| `make_tags(tag, word)` | The web is built with HTML strings like "Yay" which draws Yay as italic text. In this example, the "i" tag makes  and  which surround the word "Yay". Given tag and word strings, create the HTML string with tags around the word, e.g. "Yay". |
| `make_out_word(out, word)` | Given an "out" string, such as "<<>>", and a word, return a new string where the word is in the middle of the out string, e.g. "<<word>>". |
| `extra_end(str)` | Given a string, return a new string made of 3 copies of the last 2 chars of the original string. The string length will be at least 2. |
| `first_two(str)` | Given a string, return the string made of its first two chars, so the String "Hello" yields "He". If the string is shorter than length 2, return whatever there is, so "X" yields "X", and the empty string "" yields the empty string "". |
| `first_half(str)` | Given a string, return the first half. So the string "WooHoo" yields "Woo". |
| `without_end(str)` | Given a string, return a version without the first and last char, so "Hello" yields "ell". The string length will be at least 2. |

List 1:

| Function form | function statement |
|---|---|
| | |
| `first_last6(nums)` | Given an array of ints and/or strings, return True if 6 appears as either the first or last element in the array. The array will be length 1 or more. |
| `same_first_last(nums)` | Given an array of ints and/or strings, return True if the array is length 1 or more, and the first element and the last element are equal. |
| `make_pi(n)` | Given n in range(1,14), return a list of the first n digits of pi in order. |
| `common_end(a, b)` | Given 2 arrays of ints and/or strings, a and b, return True if they have the same first element or they have the same last element. Both arrays will be length 1 or more. |
| `sum3(nums)` | Given an array of ints, length $\geq 1$, return the sum of all the elements. |
| `rotate_left3(nums)` | Given an array of ints and/or strings, length $\geq 0$, return an array with the elements "rotated left" so [1, 2, 3] yields [2, 3, 1]. |
| `reverse3(nums)` | Given an array of ints and/or strings, length $\geq 0$, return a new array with the elements in reverse order, so [1, 2, 3] becomes [3, 2, 1]. |
| `max_end3(nums)` | Given an array of ints, length $\geq 1$, figure out which is larger, the first or last element in the array, call it $m$, and return an array of the same length where each element in the returned array is $m$. |

Logic 1:

| | |
|---|---|
| `cigar_party(`<br>`  cigars,`<br>`  is_weekend)` | When squirrels get together for a party, they like to have cigars. A squirrel party is successful when the number of cigars is between 40 and 60, inclusive. Unless it is the weekend, in which case there is no upper bound on the number of cigars. Return True if the party with the given values is successful, or False otherwise. |
| `date_fashion(`<br>`  you, date)` | You and your date are trying to get a table at a restaurant. The parameter "you" is the stylishness of your clothes, in the range 0..10, and "date" is the stylishness of your date's clothes. The result getting the table is encoded as an int value with 0=no, 1=maybe, 2=yes. If either of you is very stylish, 8 or more, then the result is 2 (yes). With the exception that if either of you has style of 2 or less, then the result is 0 (no). Otherwise the result is 1 (maybe). |
| `squirrel_play(`<br>`  temp,`<br>`  is_summer)` | The squirrels in Palo Alto spend most of the day playing. In particular, they play if the temperature is between 60 and 90 (inclusive). Unless it is summer, then the upper limit is 100 instead of 90. Given an int temperature and a boolean is_summer, return True if the squirrels play and False otherwise. |
| `caught_speeding(`<br>`  speed,`<br>`  is_birthday)` | You are driving a little too fast, and a police officer stops you. Write code to compute the result, encoded as an int value: 0=no ticket, 1=small ticket, 2=big ticket. If speed is 60 or less, the result is 0. If speed is between 61 and 80 inclusive, the result is 1. If speed is 81 or more, the result is 2. Unless it is your birthday -- on that day, your speed can be 5 higher in all cases. |
| `sorta_sum(a, b)` | Given 2 ints, a and b, return their sum. However, sums in the range 10..19 inclusive, are forbidden, so in that case just return 20. |
| `alarm_clock(`<br>`  day,`<br>`  vacation)` | Given a day of the week encoded as 0=Sun, 1=Mon, 2=Tue, ...6=Sat, and a boolean indicating if we are on vacation, return a string of the form "7:00" indicating when the alarm clock should ring. Weekdays, the alarm should be "7:00" and on the weekend it should be "10:00". Unless we are on vacation -- then on weekdays it should be "10:00" and weekends it should be "off". |
| `love6(a, b)` | The number 6 is a truly great number. Given two int values, a and b, return True if either one is 6. Or if their sum or difference is 6. Note: the function abs(num) computes the absolute value of a number. |
| `in1to10(n,`<br>`  outside_mode)` | Given a number n, return True if n is in the range 1..10, inclusive. Unless outside_mode is True, in which case return True if the number is less or equal to 1, or greater or equal to 10. |