1. (5.1) When using a security access system, different clearance levels are assigned to users. In our system, *admin* means full access, *user* means limited access, and *guest* means view-only access. Write a function named **access_rights** that takes user_role (a string) as an argument and returns the access rights of a user.

   **Examples:**

   - access_rights("user") → "limited",
   - access_rights("guest") → "view",
   - access_rights("admin") → "full"

2. (5.2) The table below show what your resting heart rate should be based on age and athleticism. Write a **function** that returns what the resting heart rate of the user should be. The arguments for the function will be *age* (how old the user is) and *athl_goal* (athletic goal of user).

   |  | Athleticism |  |
   |---|---|---|
   | Age | Above Average | Below Average |
   | 20 – 39 | 47 – 72 | 73 – 93 |
   | 40 – 59 | 46 – 71 | 72 – 94 |
   | 60 – 79 | 45 – 70 | 71 – 97 |

   **Examples:**

   - resting_rate(45, "Below Average") → "72-94",
   - resting_rate(79, "Above Average") → "45-70",
   - resting_rate(20, "Below Average") → "73-93"

3. (6.1) The **normal human body temperature** is 98.6F in Fahrenheit and 37C in Celsuis. Create a function that determines if the *temp* is considered a fever(anove normal body temperature) or not. *temp* will be measured in Fahrenheit and Celsuis.
   Notice: The F or C will always be the last character in the string.

   **Examples:**

   - is_fever("99F") → True,
   - is_fever("37C") → False,
   - is_fever("98F") → False,

1. (5.1) When using a security access system, different clearance levels are assigned to users. In our system, *admin* means full access, *user* means limited access, and *guest* means view-only access. Write a function named **access_rights** that takes user_role (a string) as an argument and returns the access rights of a user.

   **Examples:**

   - access_rights("user") → "limited",
   - access_rights("guest") → "view",
   - access_rights("admin") → "full"

2. (5.2) Write a **function** that loops through and returns the sum of all odd numbers between two integers (inclusive). The arguments to the function will be *smaller_num* and *larger_num*.

   **Examples:**

   - odd_sum(0, 7) → 16 (since 1+3+5+7 = 16)
   - odd_sum(1,10) → 25 (since 1+3+5+7+9 = 25)
   - odd_sum(50, 517) → 66456

3. (6.1) Write a function called *flip_flop* that takes a string as an argument and returns a new word made up of the second half of the word first combined with the first half of the word second.

   **Examples:**

   - *flip_flop*("abcd") → "cdab" (that is, "cd" then "ab" . . . even length)
   - *flip_flop*("grapes") → "pesgra" (that is, "pes" then "gra" . . . even length)
   - *flip_flop*("abcde")→ "decab" (that is, "de" then "c" then "ab" . . . odd length)
   - *flip_flop*("cranberries")→ "rriesecranb" (that is, "rries" then "e" then "cranb" . . . odd length)

Dark Helmet
section 4

Fctns and Strings quiz

1. (5.1) Write a **function** that returns the sum of the cubes of all positive integers up to a given integer (inclusive). The argument to the function will be *num* (the number up to which cubes should be summed).

   **Examples:**

   - cube_sum(3) → 36 ($1^3 + 2^3 + 3^3 = 36$)
   - cube_sum(8) → 1296 ($1^3 + 2^3 + 3^3 + 4^3 + 5^3 + 6^3 + 7^3 + 8^3 = 1296$)
   - cube_sum(-3) → "unknown"

2. (5.2) Write a **function** that loops through and prints every even number between two integers (inclusive). The arguments to the function will be *smaller_num* and *larger_num*.

   **Examples:**

   - output_even(37, 1050) → 38, 40, 42, . . . 1048, 1050,
   - output_even(1, 2000) → 2, 4, 6, . . . 1998, 2000
   - output_even(50, 199) → 50, 52, 54, . . . 196, 198

3. (6.1) Severus Snape seeks to harness powerful spells in the Hogwarts Library, you must encode them by using the last letter of each word. Create a function called *last_letters* that takes the variable *sentence* (a string) and returns a string made up of the last letters of each word in the sentence.

   **Examples:**

   - last_letters("wingardium leviosa makes objects float") → "masst"
   - last_letters("expecto patronum repels dementors") → "omss"
   - last_letters("the magic is within you") → "ecsnu"

1. (5.1) (Game: heads or tails) Write a **function** that lets the user guess whether the flip of a coin results in heads or tails. The function randomly generates an integer 0 or 1, which represents head or tail. The function returns if the guess is correct or incorrect. The argument for the function will be *guess* (the guess of the user, 0 for heads and 1 for tails).
   Hint: Use the following lines of code to create the function.

   ```
   from random import randint
   value = randint(0,1) #picks a random integer. Either 0 or 1.
   ```

   **Examples:**

   - toss_coin(0) → "Correct!" (if the random value is 0) or "Incorrect!" (if the random value is 1),
   - toss_coin(1) → "Correct!" (if the random value is 1) or "Incorrect!" (if the random value is 0)

2. (5.2) A toy store owner wants to know the total number of batteries needed for all the electronic toys in the store. The store sells three types of toys:

   - Electronic dolls, which require **2** batteries
   - Remote-controlled cars, which require **4** batteries
   - Robot dogs, which require **6** batteries

   Write a **function** that counts the total number of batteries needed for the toy store and returns the value. The arguments for the function will be *e_dolls* (number of electronic dolls toy store has), *rc_cars* (number of remote-controlled cars toy store has), and *robo_dogs* (number of robot dogs toy store has).

   **Examples:**

   - battery_counter(4, 3, 2) → 32,
   - battery_counter(1, 1, 1) → 12,
   - battery_counter(0, 10, 0) → 40

3. (6.1) The **boiling point** of water is 212F in Fahrenheit and 100C in Celsuis. Create a function that determines if the *temp* is considered boiling or not. *temp* will be measured in Fahrenheit and Celsuis. Notice: The F or C will always be the last character in the string.

   **Examples:**

   - is_boiling("212F") → True,
   - is_boiling("100C") → True,
   - is_boiling("0F") → False,