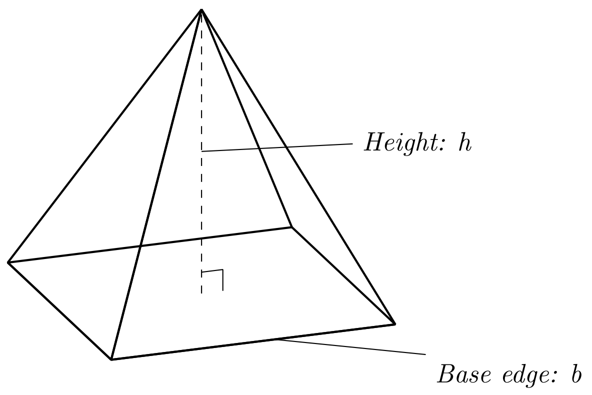
1. Write a **function** that calculates then returns the volume of a right square pyramid. The arguments of the function will be b (the base edge) and h (the height).
   Use the value of π from the math module in your calculation.
   Hint: $V = \frac{b^2h}{3}$

   Examples:

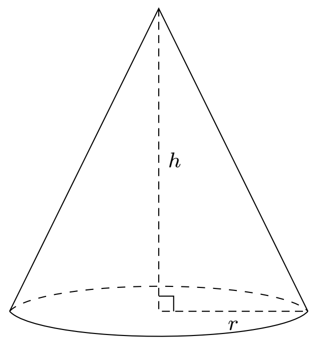
   - pyramid_volume(1, 2) $\rightarrow$ 0.66,
   - pyramid_volume(2, 2) $\rightarrow$ 2.66,
   - pyramid_volume(3, 4) $\rightarrow$ 12



2. Write a **function** that calculates then returns the volume of a cone.
   The arguments of the function will be r (the radius) and h (the height).
   Use the value of π from the math module in your calculation.
   Hint: $V = \pi \cdot \frac{r^2h}{3}$

   Examples:

   - cone_volume(1, 2) $\rightarrow$ 2.09,
   - cone_volume(3, 3) $\rightarrow$ 28.27,
   - cone_volume(1, 4) $\rightarrow$ 4.18



3. You are counting points for a basketball game. Write a **function** that calculates the final points for the team and returns the value. The arguments for the function will be *two_pointers* (number of two-pointers scored) and *three_pointers* (number of three-pointers scored).

   Examples:

   - total_score(5, 7) $\rightarrow$ 31,
   - total_score(6, 5) $\rightarrow$ 27,
   - total_score(10, 10) $\rightarrow$ 50

4. You are counting points for a tennis match. Write a **function** that calculates the total points for a player and returns the value. The arguments for the function will be *aces* (each ace is worth 2 points)

and *winning_shots* (each winning shot is worth 1 point).

Examples:

- total_score(1, 1) → 3,
- total_score(2, 3) → 7,
- total_score(5, 3) → 13

5. A farmer is asking you to tell him how many legs can be counted among all his animals. The farmer breeds three species:

   - chickens, which have **2** legs
   - cows, which have **4** legs
   - pigs, which have **4** legs

   Write a **function** that counts the total number of legs in the farmer's land and returns the value. The arguments for the function will be chickens (number of chickens farmer has), cows (number of cows farmer has), and pigs (number of pigs farmer has).

   Examples:

   - leg_counter(4, 3, 2) → 28,
   - leg_counter(1, 1, 1) → 10,
   - leg_counter(0, 10, 0) → 40

6. 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

7. 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"

8. The table below shows what time different age groups (by grade) can swim at the pool. There are two time options, morning and afternoon. Write a **function** that returns the time the pool is available for the user. The arguments for the function will be *grade* (which grade the user is in) and *time* (which time the user would like to go in).

	Pool times	
Grade	Morning	Afternoon
k, 1 – 3	9 AM	1 PM
4 – 8	10 AM	2 PM
9 – 12	11 AM	3 PM

Examples:

- pool_time('k', "Morning") → "9 AM",
- pool_time(5, "Afternoon") → "2 PM",
- pool_time(12, "Morning") → "11 AM"

9. When driving in a car and approaching a traffic control light, *green* means go, *yellow* means yield, and *red* means stop. Write a **function** that returns the action that should be taken by a driver. The argument for the function will be *light_color* (the current color of the traffic control light).

Examples:

- traffic_light("red") → "Stop",
- traffic_light("Green") → "Go",
- traffic_light("Yellow") → "Yield"

10. 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"

11. In Harry Potter, the currency consists of knuts, sickle, and galleon. There are 29 knuts in one sickle and 17 sickles in one galleon. Write a **function** that will return a converted amount of knuts into the fewest amount of coins possible. Only return a string with the non-zero values, meaning don't return something similar to "0 sickles". The argument for the function will be *knuts* (how many knuts to convert).

Examples:

- convert_knuts(32) → "1 sickle 3 knuts",
- convert_knuts(544) → "1 galleon 4 sickles 18 knuts",

- convert_knuts(993) → "2 galleons 7 knuts"
  Note: Do **not** output 2 galleons 0 sickle 7 knuts.

12. In an Ancient Kingdom, the currency consists of bronze coins, silver coins, and gold coins. There are 20 bronze coins in one silver coin and 15 silver coins in one gold coin. Write a **function** that will return a converted amount of bronze coins into the fewest amount of coins possible. Only return a string with the non-zero values, meaning don't return something similar to "0 silver coins". The argument for the function will be *bronze_coins* (how many bronze coins to convert).

    Examples:

    - convert_bronze(32) → "1 silver 12 bronze",
    - convert_bronze(544) → "1 gold 4 silver 4 bronze",
    - convert_bronze(903) → "3 gold 3 bronze"
      Note: Do **not** output 3 gold 0 silver 3 bronze.

13. (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)

14. (Game: Odd or Even) Write a **function** that lets the user guess whether a randomly generated number is odd or even. The function randomly generates an integer between 0 and 9 (inclusive) and returns whether the user's guess is correct or incorrect. The argument for the function will be *guess* (the user's guess, either "odd" or "even").
    Hint: Use the following lines of code to create the function.

    ```
    from random import randint
    value = randint(0,9) #picks a random integer between 0-9 inclusive
    ```

    Examples:

    - guess_num("odd") → "Correct!" (if the random value is odd) or "Incorrect!" (if the random value is even),
    - guess_num("even") → "Correct!" (if the random value is even) or "Incorrect!" (if the random value is odd)

15. Write a **function** that returns the number of copies of the same number. The arguments for the function will be *num_1* (first number), *num_2* (second number), and *num_3* (third number).

    Examples:

    - count_duplicates(2, 3, 2) → "You entered the same number 2 times",

- count_duplicates(4, 4, 4) → "You entered the same number 3 times",
- count_duplicates(1, 2, 3) → "Each number is unique"

16. Primary U.S. interstate highways are numbered 1-99. Odd numbers (like 5 or 95) go north/ south, and evens (like 10 or 82) go east/west. Auxiliary highways are numbered 100-999, and service the primary highway indicated by the rightmost two digits. Thus, I-405 services I-5, and I-290 services I-90.

    Note: 200 is not a valid auxiliary highway because 00 is not a valid primary highway number.

    Write a **function** that returns whether the highway runs north/south or east/west or is an invalid highway number. The argument for the function will be *highway_num*(highway number provided).
    Examples:

    - highway_directions(5) → "I-5 runs north/south",
    - highway_directions(82) → "I-82 runs east/west",
    - highway_directions(200) → "I-200 is an invalid highway number"

17. Write a **function** that checks whether a letter is a vowel or consonant. The function should return "Vowel" if the letter is a vowel and "Consonant" if the letter is a consonant. The argument to the function will be *letter* (a single lowercase letter).
    Hint: In the English language, the letters a, e, i, o, and u are the vowels.
    Examples:

    - check_letter('a') → "Vowel",
    - check_letter('z') → "Consonant",
    - check_letter('o') → "Vowel"

18. At the local ice cream store they have 3 flavors, which are Vanilla, Chocolate, and Strawberry. Write a **function** that returns the selected flavor. If the chosen flavor is not available, let them know. The argument to the function will be $selected\_flavor$ (the user's selected flavor).
    Examples:

    - serve_icecream("Vanilla") → "Here is your vanilla ice cream!",
    - serve_icecream("Mint") → "Sorry, we don't have mint ice cream.",
    - serve_icecream("Chocolate") → "Here is your chocolate ice cream!"

19. At the local coffee shop they have 3 types of coffee, which are Espresso, Latte, and Cappuccino. Write a **function** that returns the selected coffee type. If the chosen type is not available, let them know. The argument to the function will be $selected\_coffee$ (the user's selected coffee type).
    Examples:

    - serve_coffee("Latte") → "Here is your latte!",
    - serve_coffee("Mocha") → "Sorry, we don't have mocha.",
    - serve_coffee("Espresso") → "Here is your espresso!"

20. Write a **function** to create a game of Rock, Paper, Scissors. The function will return the winner of the game played by two players. The arguments to the function will be $player1$ (the first player's choice) and $player2$ (the second player's choice).
    Print the winner according to the following rules.

    - Rock beats Scissors
    - Scissors beats Paper

- Paper beats Rock

Examples:

- find_winner("Rock", "Paper") → "Player 2 wins!",
- find_winner("Scissors", "Paper") → "Player 1 wins!",
- find_winner("Rock", "Rock") → "It's a tie!"

21. Write a **function** that returns if a triangle is a scalene, isosceles, or an equilateral. The arguments to the function will be *side_1* (first side of the triangle), *side_2* (second side of the triangle), and *side_3* (third side of the triangle).
    The types of triangles are:

    - No sides equal: "scalene"
    - Two sides equal: "isosceles"
    - All sides equal: "equilateral"

    Examples:

    - triangle_type(1, 2, 3) → "scalene",
    - triangle_type(4, 4, 4) → "equilateral",
    - triangle_type(7, 7, 3) → "isosceles"

22. Luke Skywalker has friends and family, but he is getting older and having trouble remembering them all. Write a **function** that will return the relation defined in the table below. The arguments to the function will be *name* (name of the person related to Luke).

Person	Relation
Darth Vader	Father
Leia	Sister
Han	Brother in law
R2D2	Droid

*If he types any other name, return "unknown".

Examples:

- find_relation("Darth Vader") → "Father",
- find_relation("R2D2") → "Droid",
- find_relation("Jabba the Hutt") → "Unknown"

23. Detective Sherlock Holmes is solving cases, but he sometimes forgets the key suspects in his investigations. Write a **function** that will return the suspect defined in the table below. The arguments to the function will be *name* (name of the suspect).

Person	Role
Moriarty	Archenemy
Watson	Best Friend
Mrs. Hudson	Landlady
Inspector Lestrade	Detective

*If he types any other name, return "unknown".

Examples:

- find_relation("Moriarty") → "Archenemy",
- find_relation("Mrs. Hudson") → "Landlady",
- find_relation("Mycroft Holmes") → "Unknown"

24. Write a **function** that loops through a word and prints every other letter of the word starting with the **second** letter. The argument to the function will be *word* (the word to loop through and output every other letter).

    Examples:

    - skip_letter("counterattack") → "oneatc",
    - skip_letter("banana sunday") → "aaasna"

25. Write a **function** that loops through a word and prints every other letter of the word starting with the **first** letter. The argument to the function will be *word* (the word to loop through and output every other letter).

    Examples:

    - skip_letter("counterattack") → "cutrtac",
    - skip_letter("banana sunday") → "bnnsna"

26. 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

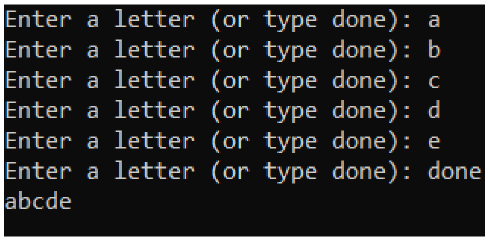
27. Write a **function** that loops through and returns the sum of all odd numbers between two integers (inclusive). The arguments to the function will be *smaller_num* and *larger_num*.

    Examples:

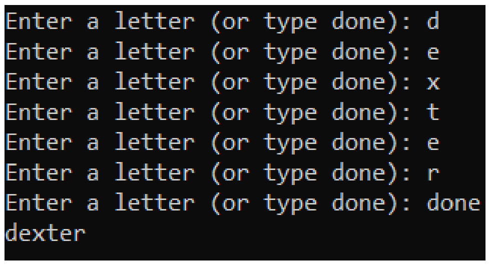
    - odd_sum(0, 7) → 16 (since 1+3+5+7 = 16)
    - odd_sum(1,10) → 25 (since 1+3+5+7+9 = 25)
    - odd_sum(50, 517) → 66456

28. Write a **function** that creates a word one letter at a time until the user types done, and returns the newly created word. There are no arguments to this function. You may name the function *create_word*().

    Examples:



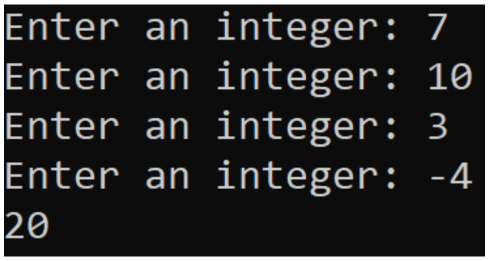
```
Enter a letter (or type done): a
Enter a letter (or type done): b
Enter a letter (or type done): c
Enter a letter (or type done): d
Enter a letter (or type done): e
Enter a letter (or type done): done
abcde
```



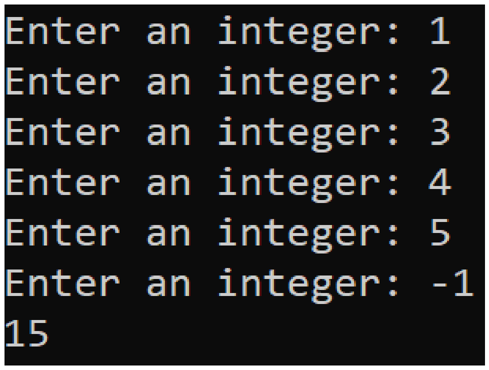
```
Enter a letter (or type done): d
Enter a letter (or type done): e
Enter a letter (or type done): x
Enter a letter (or type done): t
Enter a letter (or type done): e
Enter a letter (or type done): r
Enter a letter (or type done): done
dexter
```

29. Write a **function** that sums up integers until a negative integer is given and then returns the sum. There are no arguments to this function. You may name the function *sum_loop*().

    Examples:



```
Enter an integer: 7
Enter an integer: 10
Enter an integer: 3
Enter an integer: -4
20
```



```
Enter an integer: 1
Enter an integer: 2
Enter an integer: 3
Enter an integer: 4
Enter an integer: 5
Enter an integer: -1
15
```

30. Given a positive integer n, the following rules will always create a sequence that ends with 1, called Hailstone Sequence:

    (a) If n is even, divide by 2

    (b) If n is odd, multiply by 3 and add 1 (i.e. $3n + 1$)

    (c) Continue until n is 1

    Write a **function** that returns the hailstone sequence starting at n. The argument to the function will be n (the integer to start the sequence from). **Examples:**

    - hailstone_seq(25) → 25, 76, 38, 19, 58 ... 8, 4, 2, 1,
    - hailstone_seq(40) → 40, 20, 10, 5, 16, 8, 4, 2, 1

31. Write a **function** that returns the factors of a given integer. The argument of the function will be *num* (integer to find factors for).

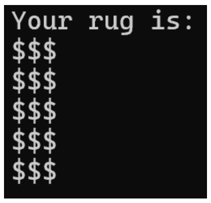
    Examples:

    - find_factors(12) → 1, 2, 3, 4, 6, 12,
    - find_factors(17) → 1, 17,
    - find_factors(36) → 1, 2, 3, 4, 6, 9, 12, 18, 36

32. You are the newest rug fashion designer on the scene, but you're running out of ideas. Write a **function** that will help you design rugs. The function will return a formatted string that will resemble a designed rug. The first parameter must be *width* (how wide the rug will be), the second must be *length* (how long the rug will be), and the third must be *pattern* (the character pattern used in the rug design).

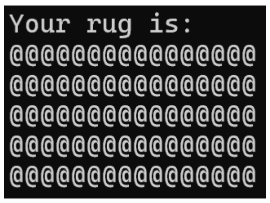
    Examples:

design_run(3,5,$) →



```
Your rug is:
$$$
$$$
$$$
$$$
$$$
```

design_run(16,5,) →



```
Your rug is:
@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@
```

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

    Examples:

    - square_sum(3) $\rightarrow$ 14 ($1^2 + 2^2 + 3^2 = 14$)
    - square_sum(8) $\rightarrow$ 204 ($1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2 + 7^2 + 8^2 = 204$)
    - square_sum(-3) $\rightarrow$ "unknown"

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