

## Exercises

1. **List four phases of the software development process, and explain what they accomplish.**

The waterfall model consists of several phases:

1. **Customer request**—In this phase, the programmers receive a broad statement of a problem that is potentially amenable to a computerized solution. This step is also called the user requirements phase.
2. **Analysis**—The programmers determine what the program will do. This is sometimes viewed as a process of clarifying the specifications for the problem.
3. **Design**—The programmers determine how the program will do its task.
4. **Implementation**—The programmers write the program. This step is also called the coding phase.
5. **Integration**—Large programs have many parts. In the integration phase, these parts are brought together into a smoothly functioning whole, usually not an easy task.
6. **Maintenance**—Programs usually have a long life; a life span of 5 to 15 years is common for software. During this time, requirements change, errors are detected, and minor or major modifications are made.

2. **Jack says that he will not bother with analysis and design but proceed directly to coding his programs. Why is that not a good idea?**

Jack is probably wrong with his approach. It might work just fine when coding simple programs like those in the previous exercises but will for sure not when working on anything long. Writing the needed constants and variables beforehand, thinking of the functions the program will use and even drawing some flow chart on a piece of paper. Of course, the more complex the program, the more elaborate the preparation phase has to be and the more pronounced those software development phases will be. Jack may succeed with this approach but it will probably take him longer and may cost him more money as he might need to rewrite parts of his program multiple times or even throw the whole thing away.

3. Let the variable x be "dog" and the variable y be "cat". Write the values returned by the following operations:

- a.  $x + y$
- b. "the " + x + " chases the " + y
- c.  $x * 4$

```
0s  x="dog"
y="cat"
print(x+y)
print("the " + x + " chase the " + y)
print(x*4)
```

dogcat  
the dog chase the cat  
dogdogdogdog

4. Write a string that contains your name and address on separate lines using embedded newline characters. Then write the same string literal without the newline characters.

A:-

```
0s  print("salem\n dubai hills St,\n maple3B\n dubai\n UAE")
```

salem  
dubai hills St,  
maple3B  
dubai  
UAE

Please note we had to use the print() function in order to process both the \n escape sequence and the multiline string.

B:-

```
0s  print("salem,Dubai hills St,maple3B,Dubai,UAE")
```

salem,Dubai hills St,maple3B,Dubai,UAE

### 5. How does one include an apostrophe as a character within a string literal?

You have to either use a double quotes or escape that single quote:

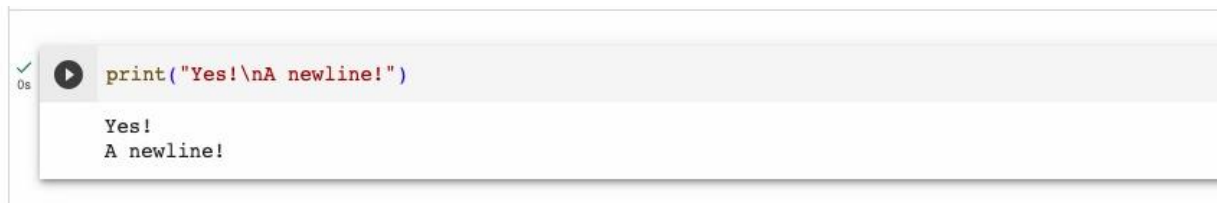


```
print("saalem's great")
print('saalem\'s great')
```

saalem's great  
saalem's great

### 6. What happens when the print function prints a string literal with embedded newline characters?

It prints also the newline characters which means there will be a new line printed in the shell. See below:



```
print("Yes!\nA newline!")
```

Yes!  
A newline!

### 7. Which of the following are valid variable names?

- a. length
- b. \_width
- c. firstBase
- d. 2MoreToGo
- e. halt!

As you can see, only the choices **a.**, **.b.** and **.c.** are correct as the two remainings will cause aSyntaxError.

**8. List two of the purposes of program documentation.**

A good program documentation is one of the most important things for its maintenance and shareability. You might know what exactly is each function or variable doing a month after the program is completed. But you sure will forget after a year. And remember that this is not limited just to you! As you might share the software with someone in the future, a good documentation will help them understand how it's designed and how does it work.

So those two purposes would be:

- Increase the program readability to make the maintenance easier.
- Explain the program function to other programmers.

**9. Which data type would most appropriately be used to represent the following data values?**

- a. The number of months in a year
- b. The area of a circle
- c. The current minimum wage
- d. The approximate age of the universe (12,000,000,000 years)
- e. Your name

a. The number of months in a year:

Data Type: **Integer (int)**

Explanation: The number of months in a year is a whole number, and integers are used to represent whole numbers in programming.

b. The area of a circle:

Data Type: **Floating-point (float)**

Explanation: The area of a circle can be a decimal or fractional number, and floating-point data types are used to represent real numbers with decimal points.

c. The current minimum wage:

Data Type: **Floating-point (float)**

Explanation: Minimum wage is typically a decimal value, and using a floating-point type is appropriate to represent monetary values.

d. The approximate age of the universe (12,000,000,000 years):

Data Type: **Integer (int)**

Explanation: While the age of the universe is a large number, it's typically represented as an integer or long integer, depending on the programming language, to ensure precision.

e. Your name:

Data Type: **String (str)**

Explanation: Names are usually represented as text, and strings are the appropriate data type for handling textual information in programming.

### 10. Explain the differences between the data types int and float.

**Integer (int)** and **float float** are both numerical data types and both can store whole numbers (positive, zero or negative), but the main difference is that **float can also represent real numbers float can also represent real numbers** (with decimal point). Generally, float is commonly used with decimal notation (e.g. 5.43), but in some cases when working with enormous numbers it can be represented by scientific notation (e.g. 5.43e0). In addition, it is worth mentioning that float data type has limited precision of approximately 16 digits.

### 11. Write the values of the following floating-point numbers in Python's scientific notation:

- a. 355.76
- b. 0.007832
- c. 4.3212

The values are:

- a. 3.5576e2
- b. 7.832e-3
- c. 4.3212e0

The way the numbers are represented are just a version of the mathematic scientific notation:

$$n.m \times 10^y \quad (1)$$

So for example, in the first number, 355.76 will get translated to  $n = 3$ ,  $m = 5576$  and as we have to divide the number by 100 ( $= 10^2$ ), the  $y$  value will be 2.

The same applies for second and third part of the assignment, only with a different exponent values. For the second task, the value is less than zero meaning we have to multiply the number by 1000 ( $= 10^{-3}$ ).

As the third number is already just a single digit before the decimal point, we do not have to divide or multiply ( $= \times 1$ ) it resulting in zero exponent as  $10^0 = 1$ .

**12. Consult Table 2-5 to write the ASCII values of the characters '\$' and '&'.**

Character	ASCII value
\$	36
&	38


You may see the characters and their ASCII table values in the table.

Table 2-5 shows the mapping of character values to the first 128 ASCII codes. The digits in the left column represent the leftmost digits of an ASCII code, and the digits in the top row are the rightmost digits. Thus, the ASCII code of the character 'R' at row 8, column 2 is 82.

	0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
1	LF	VT	FF	CR	SO	SI	DLE	DCI	DC2	DC3
2	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
3	RS	US	SP	!	"	#	\$	%	&	'
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	P	q	r	S	t	u	v	w
12	X	y	z	{		}	~	DEL		

**13. Let  $x=8$  and  $y=2$ . Write the values of the following expressions:**

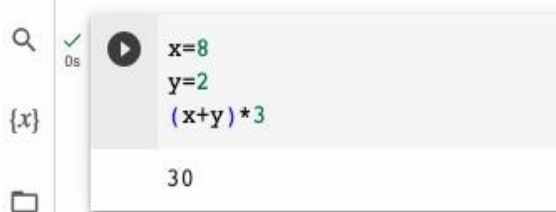
a.  $x + y * 3$



A screenshot of a Python REPL interface. On the left, there is a search icon, a green checkmark with '0s', a '{x}' symbol, and a folder icon. The main area shows a code block with a play button icon, followed by the assignments `x=8`, `y=2`, and the expression `x+y*3`. Below the code block, a box displays the result `14`.

The  $y*3$  part gets computed first due to operator priority, then the result is added to the  $x$  value.

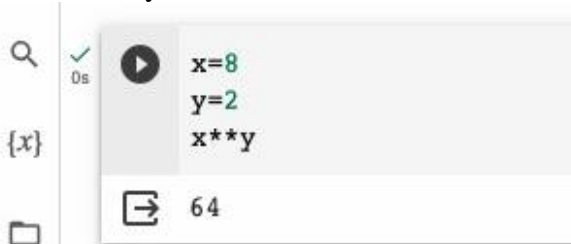
b.  $(x + y) * 3$



A screenshot of a Python REPL interface. On the left, there is a search icon, a green checkmark with '0s', a '{x}' symbol, and a folder icon. The main area shows a code block with a play button icon, followed by the assignments `x=8`, `y=2`, and the expression `(x+y)*3`. Below the code block, a box displays the result `30`.

First, we compute the inner part of the parentheses as they have a higher priority, then we multiply the result by 3.

c.  $x ** y$



A screenshot of a Python REPL interface. On the left, there is a search icon, a green checkmark with '0s', a '{x}' symbol, and a folder icon. The main area shows a code block with a play button icon, followed by the assignments `x=8`, `y=2`, and the expression `x**y`. Below the code block, a box displays the result `64`.

Here we simply use exponentiation. This could be also written as  $X^y$

d.  $x \% y$



```
x=8
y=2
x*y
0
```

The  $\%$  operator uses a division with a remainder, giving us the remainder. As the numbers may be divided without a remainder, the result is zero.

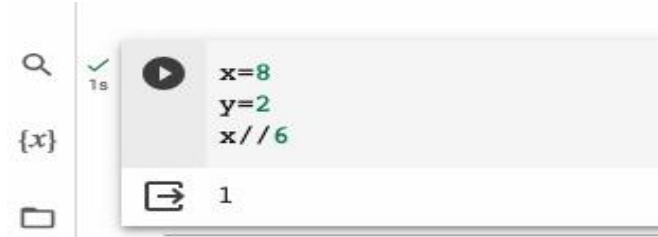
e.  $x / 12.0$



```
x=8
y=2
x/12.0
0.6666666666666666
```

A simple arithmetic division.

f.  $x // 6$



```
x=8
y=2
x//6
1
```

The  $//$  operator will output the division with a remainder result.

#### 14. Let $x = 4.66$ Write the values of the following expressions:

a. `round(x)`

b. `int(x)`

A:-





```
x=4.66
round(x)
```

5

As you can see, the the `round()` function with the number as an argument, will mathematically round the given number to the nearest integer.

B:-




```
x=4.66
int(x)
```

4

`int()` type-casting will just cut out the whole part of the number.

### 15. How does a Python programmer round a float value to the nearest int value?

A Python programmer can round a float value to the nearest integer using the **`round()`** function. The **`round()`** function takes a floating-point number as input and returns the nearest integer. Here's how you can use it:



```
float_value=8.9
round(8.9)
```

9

### 16. How does a Python programmer concatenate a numeric value to a string value?

We may use the `+` operator but we have to cast the numeric value to a **string** first.



```
str(100)+"% salem"
```

'100% salem'

Or, **we can simply use a comma (,)** when we're printing the values out (but that is not a concatenation *per se* as we're passing a list of values to the `print()` function instead.



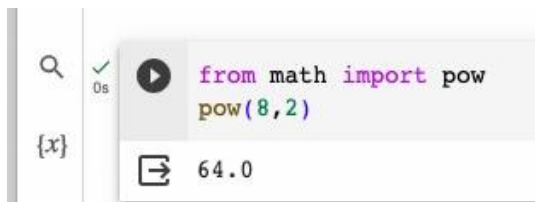
A screenshot of a code editor interface. On the left, there is a search icon and a checkmark. The code editor shows a single line of Python code: `print(100, "% salem")`. Below the code, the output is displayed as `100 % salem`. To the right of the output, there are two buttons: "+ Code" and "+".

### 17. Explain the relationship between a function and its arguments.

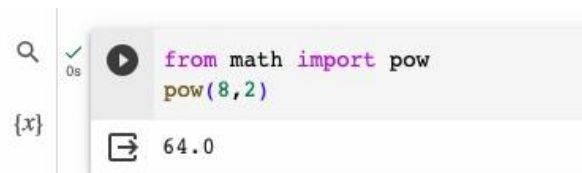
In Python, a function is a block of reusable code that performs a specific task or a set of tasks. Functions are defined using the `def` keyword and are typically designed to take input values, called arguments, and produce an output or perform some action based on those arguments.

The relationship between a function and its arguments is essential for creating flexible and reusable code. Functions enable you to abstract away the implementation details and work with specific data as needed, promoting code organization and reusability.

### 18. The `math` module includes a `pow` function that raises a number to a given power. The first argument is the number, and the second argument is the exponent. Write a code segment that imports this function and calls it to print the values $8^2$ and $5^4$ .



A screenshot of a code editor interface. On the left, there is a search icon and a checkmark. The code editor shows two lines of Python code: `from math import pow` and `pow(8,2)`. Below the code, the output is displayed as `64.0`.

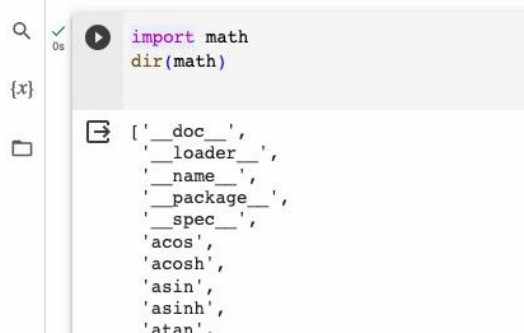


A screenshot of a code editor interface. On the left, there is a search icon and a checkmark. The code editor shows two lines of Python code: `from math import pow` and `pow(8,2)`. Below the code, the output is displayed as `64.0`.

The `pow()` function returns a float, so the results are displayed as floating-point numbers. If you want integer results, you can use the `int()` function to convert them to integers.

### 19. Explain how to display a directory of all of the functions in a given module

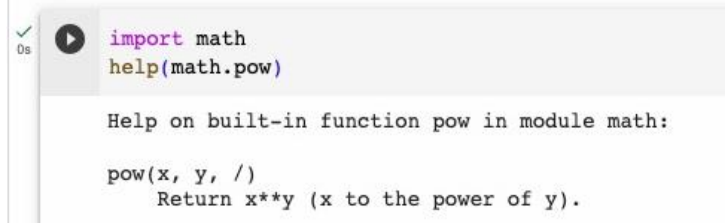
Example below for math module



```
import math
dir(math)
```

```
['_doc_', '_loader_', '_name_', '_package_', '_spec_', 'acos', 'acosh', 'asin', 'asinh', 'atan']
```

### 20. Explain how to display help information on a particular function in a given module.



```
import math
help(math.pow)
```

```
Help on built-in function pow in module math:

pow(x, y, /)
    Return x**y (x to the power of y).
```

## Review Questions

### 1. What does a programmer do during the analysis phase of software development?

- a. Codes the program in a particular programming language
- b. Writes the algorithms for solving a problem
- c. Decides what the program will do and determines its user interface
- d. Tests the program to verify its correctness.

In the analysis phase, the focus is on understanding and defining the requirements of the software. This includes determining the functionality, features, and user interface of the software based on the needs of the end-users and the problem the software aims to solve. It's about defining "what" the software needs to do, not "how" it will be implemented (which comes later during the design and coding phases).

The other options mentioned are activities that typically occur in later phases of software development.

### 2. What must a programmer use to test a program?

- a. All possible sets of legitimate inputs
- b. All possible sets of inputs
- c. A single set of legitimate inputs
- d. A reasonable set of legitimate inputs

The correct answer is **d. A reasonable set of legitimate inputs**. The other choices would also kind of work but we won't get the required results by using them. With both **a.** and **b.** it may take us infinitely long to do such testing, and with **c.** the testing would not be thorough enough.

**3. What must you use to create a multi-line string?**

- a. A single pair of double quotation marks
- b. A single pair of single quotation marks
- c. A single pair of three consecutive double quotation marks
- d. Embedded newline characters

**4. What is used to begin an end-of-line comment?**

- a. / symbol
- b. # symbol
- c. % symbol

Anything following the # symbol on a line is considered a comment and is not executed as code.

**5. Which of the following lists of operators is ordered by decreasing precedence?**

- a. +, \*, \*\*
- b. \*, /, %
- c. \*\*, \*, +

The precedence of operators in Python follows the PEMDAS (Parentheses, Exponents, Multiplication and Division, Addition and Subtraction) rule, where "\*\*\*" (exponentiation) has the highest precedence, followed by "\*", (multiplication) and then "+", (addition) with multiplication and addition having equal precedence and evaluated from left to right.

**6. The expression `2 ** 3 ** 2` evaluates to which of the following values?**

- a. 64
- b. 512
- c. 8

Test it out easily in Python

The expression `2 ** 3 ** 2` is evaluated right to left, following the operator precedence rules in Python. So, it is equivalent to `2 ** (3 ** 2)`.

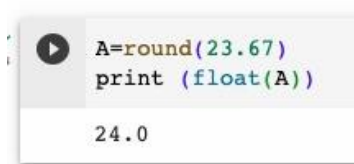
First, `3 ** 2` equals 9

Then, `2 ** 9` equals 512.

So, the expression `2 ** 3 ** 2` evaluates to b. 512

7. The expression `round(23.67)` evaluates to which of the following values?

- a. 23
- b. 23.7
- c. 24.0



```
A=round(23.67)
print(float(A))
```

24.0

You can change it to float if you want the exact number.

8. Assume that the variable `name` has the value 33. What is the value of `name` after the assignment `name = name * 2` executes?

- a. 35
- b. 33
- c. 66



```
name=33
name=name*2
print(name)
```

66

Or



```
name=33
name*2
```

66

9. Write an import statement that imports just the functions `sqrt` and `log` from the `math` module.



A code editor snippet with a search icon, a green checkmark, and a play button. The code is:

```
from math import log
from math import sqrt
```

Or



A code editor snippet with a search icon, a green checkmark, and a play button. The code is:

```
from math import log, sqrt
```

10. What is the purpose of the `dir` function and the `help` function?

The **`dir`** function allows us to see directory of all available functions of an imported module. The **`help`** function prints out the docstrings of function or functions of a module.

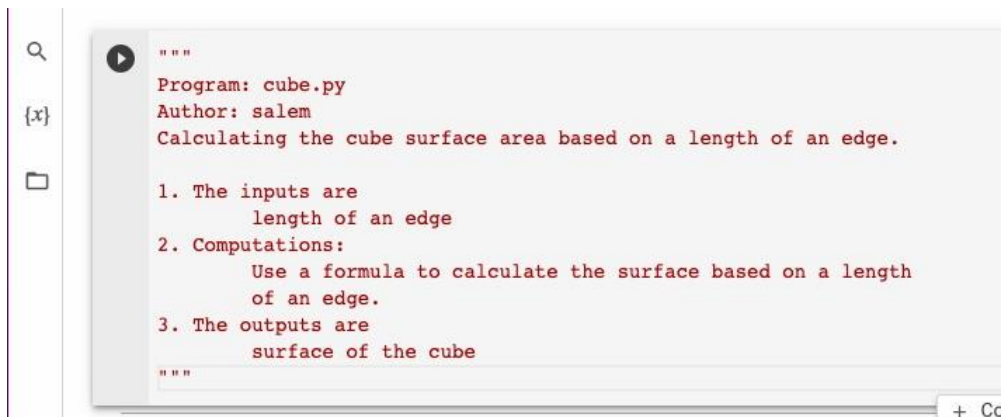
## Projects

In each of the projects that follow, you should write a program that contains an introductory docstring. This documentation should describe what the program will do (analysis) and how it will do it (design the program in the form of a pseudocode algorithm). Include suitable prompts for all inputs, and label all outputs appropriately. After you have coded a program, be sure to test it with a reasonable set of legitimate inputs.

1. You can calculate the surface area of a cube if you know the length of an edge. Write a program that takes the length of an edge (an integer) as input and prints the cube's surface area as output.

You can calculate the surface area of a cube using the formula:  $\text{Surface Area} = 6 * (\text{Edge Length})^2$

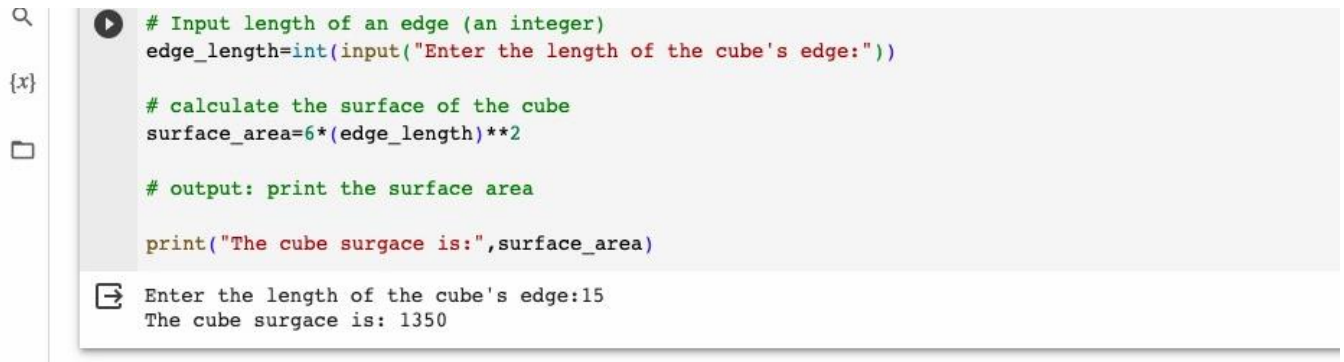
Step one wiring the introductory docstring



```
"""
Program: cube.py
Author: salem
Calculating the cube surface area based on a length of an edge.

1. The inputs are
    length of an edge
2. Computations:
    Use a formula to calculate the surface based on a length
    of an edge.
3. The outputs are
    surface of the cube
"""
```

Step two writing the program below.



```
# Input length of an edge (an integer)
edge_length=int(input("Enter the length of the cube's edge:"))

# calculate the surface of the cube
surface_area=6*(edge_length)**2

# output: print the surface area

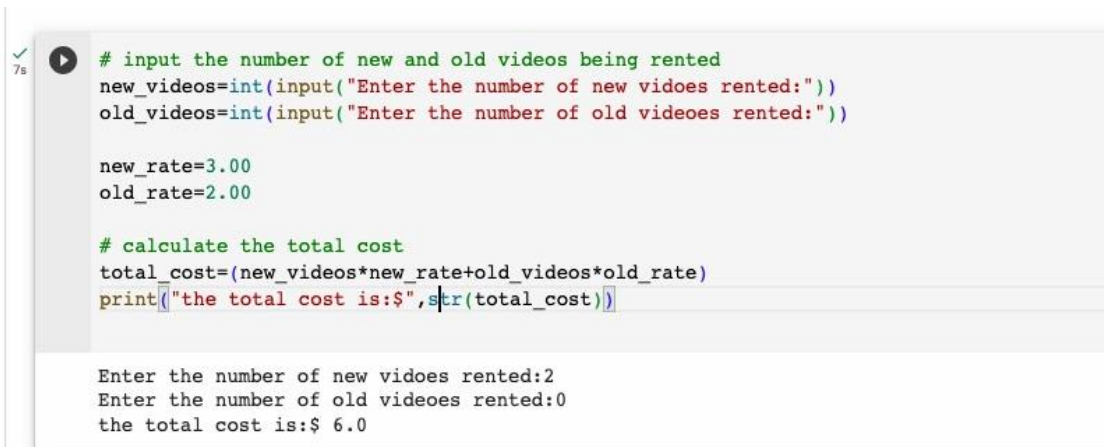
print("The cube surgace is:",surface_area)
```

Enter the length of the cube's edge:15  
The cube surgace is: 1350

2. Five Star Retro Video rents VHS tapes and DVDs to the same connoisseurs who like to buy LP record albums. The store rents new videos for \$3.00 a night, and oldies for \$2.00 a night. Write a program that the clerks at Five Star Retro Video can use to calculate the total charge for a customer's video rentals. The program should prompt the user for the number of each type of video and output the total cost.

Step one wiring the introductory docstring same as the above question.

Step two writing the program.



```
# input the number of new and old videos being rented
new_videos=int(input("Enter the number of new vidoes rented:"))
old_videos=int(input("Enter the number of old vidoes rented:"))

new_rate=3.00
old_rate=2.00

# calculate the total cost
total_cost=(new_videos*new_rate+old_videos*old_rate)
print("the total cost is:$",str(total_cost))
```

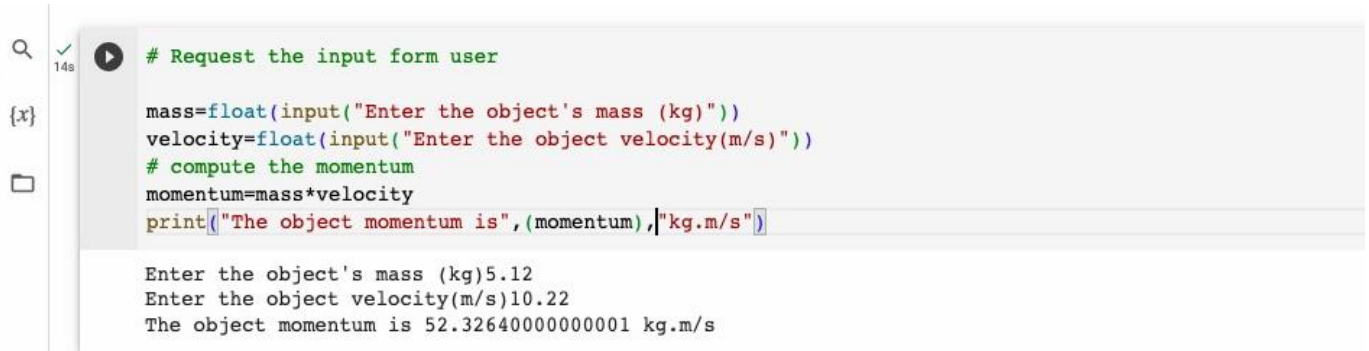
Enter the number of new vidoes rented:2  
Enter the number of old vidoes rented:0  
the total cost is:\$ 6.0



3. Write a program that takes the radius of a sphere (a floating-point number) as input and then outputs the sphere's diameter, circumference, surface area, and volume.

Step one wiring the introductory docstring same as the above question.

Step two writing the program.

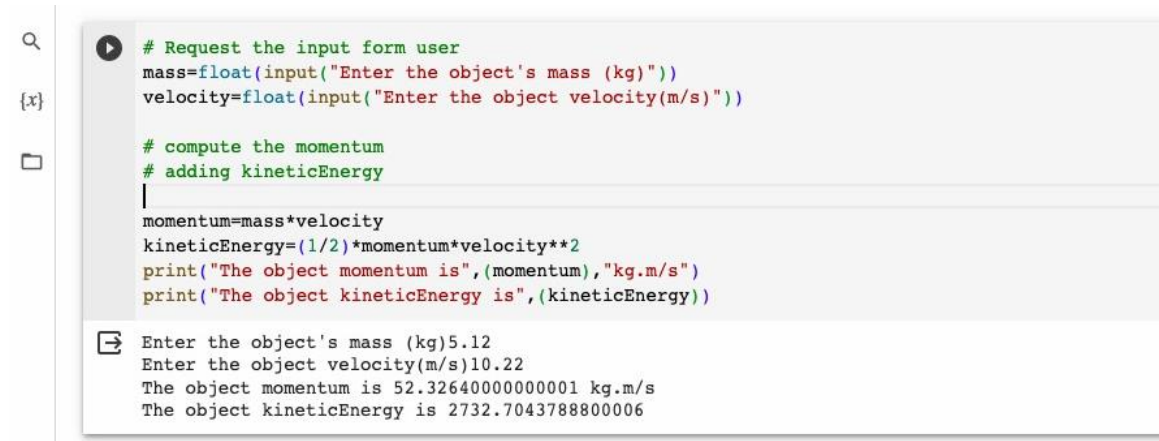


```
# Request the input form user

mass=float(input("Enter the object's mass (kg)"))
velocity=float(input("Enter the object velocity(m/s)"))
# compute the momentum
momentum=mass*velocity
print("The object momentum is", (momentum), "kg.m/s")
```

Enter the object's mass (kg)5.12  
Enter the object velocity(m/s)10.22  
The object momentum is 52.32640000000001 kg.m/s

4. The kinetic energy of a moving object is given by the formula  $KE = (1/2) mv^2$  where  $m$  is the object's mass and  $v$  is its velocity. Modify the program you created in Project 5 so that it prints the object's kinetic energy as well as its momentum.



```
# Request the input form user
mass=float(input("Enter the object's mass (kg)"))
velocity=float(input("Enter the object velocity(m/s)"))

# compute the momentum
# adding kineticEnergy
|
momentum=mass*velocity
kineticEnergy=(1/2)*momentum*velocity**2
print("The object momentum is", (momentum), "kg.m/s")
print("The object kineticEnergy is", (kineticEnergy))
```

Enter the object's mass (kg)5.12  
Enter the object velocity(m/s)10.22  
The object momentum is 52.32640000000001 kg.m/s  
The object kineticEnergy is 2732.7043788800006

5. Write a program that calculates and prints the number of minutes in a year.

```
0s
# mins cal
MINS_IN_HOUR = 60
HOURS_IN_DAY = 24
DAYS_IN_YEAR = 365
# Compute the number of minutes
minutes = MINS_IN_HOUR * HOURS_IN_DAY * DAYS_IN_YEAR
# Display the number of minutes in a day
print("The number of minutes in a year is:", minutes)
```

The number of minutes in a year is: 525600

6. Light travels at  $3 \times 10^8$  meters per second. A light-year is the distance a light beam travels in one year. Write a program that calculates and displays the value of a light-year.

```
# Initialize the constants
SPEED_OF_LIGHT = 3e8
SECS_IN_MIN = 60
MINS_IN_HOUR = 60
HOURS_IN_DAY = 24
DAYS_IN_YEAR = 365
# Compute the number of meters traveled by light in a year
meters = SPEED_OF_LIGHT * SECS_IN_MIN * MINS_IN_HOUR * HOURS_IN_DAY * DAYS_IN_YEAR
# Display number of meters
print("The light travels", round(meters), "meters in a year.")
```

The light travels 9460800000000000 meters in a year.

**7. Write a program that takes as input a number of kilometers and prints the corresponding number of nautical miles. Use the following approximations:**

- A kilometer represents 1/10,000 of the distance between the North Pole and the equator
- There are 90 degrees, containing 60 minutes of arc each, between the North Pole and the equator.
- A nautical mile is 1 minute of an arc.

```
0s # Initialize the constants
SPEED_OF_LIGHT = 3e8
SECS_IN_MIN = 60
MINS_IN_HOUR = 60
HOURS_IN_DAY = 24
DAYS_IN_YEAR = 365

# Compute the number of meters traveled by light in a year
meters = SPEED_OF_LIGHT * SECS_IN_MIN * MINS_IN_HOUR * HOURS_IN_DAY * DAYS_IN_YEAR
# Display number of meters
print("The light travels", round(meters), "meters in a year.")

The light travels 9460800000000000 meters in a year.
```

**8. An employee's total weekly pay equals the hourly wage multiplied by the total number of regular hours plus any overtime pay. Overtime pay equals the total overtime hours multiplied by 1.5 times the hourly wage. Write a program that takes as inputs the hourly wage, total regular hours, and total overtime hours and displays an employee's total weekly pay.**

```
14s # Initialize the constants
REGULAR_MULTIPLIER = 1
OVERTIME_MULTIPLIER = 1.5

# Request the inputs
hourlyWage = float(input("Enter the wage: "))
regular = float(input("Enter the regular hours: "))
overtime = float(input("Enter the overtime hours: "))

# Compute the wages
regular_wage = regular * REGULAR_MULTIPLIER
overtime_wage = overtime * OVERTIME_MULTIPLIER
total_wage = hourlyWage * (regular_wage + overtime_wage)

# Display the total wage
print("The employee total wage is", round(total_wage, 2))

Enter the wage: 10
Enter the regular hours: 12
Enter the overtime hours: 13
The employee total wage is 315.0
```