1)

Operators : \* - + /

Values : ‘hello’ -88.8 5

2)

Variable : spam

String : ‘spam’

3)

Integer, String, Boolean

4)

An expression consists of values and operators and always evaluates down to a single value.

5)

An expression is always evaluated. Produces or returns a value as a result. A statement's execution may or may not create or show a result value, it just accomplishes what the statement specifies.

6)

Bacon = 20

Bacon + 1 will result in 21 but bacon value isn’t changed, it’s still 20. To change it to 21, the code should be bacon = bacon + 1 or bacon += 1

7)

‘Spam’ + ‘spamspam’ = ‘spamspamspam’

‘Spam’ \* 3 = ‘spamspamspam’

8)

Because 100 is a number, numbers can’t be variables.

9)

int(), str(), float()

10)

‘I have eaten ‘ + 99 + ‘ burritos.’ Causes an error because it’s trying to add a number into a string. To fix it, we have to put an apostrophe around number 99

‘I have eaten ‘ + ‘99’ + ‘ burritos’ is the correct answer

Extra : The round() method produces a floating point value with the provided number of decimals that is a rounded version of the specified integer.

1)

True and False

2)

And, Or, Not

3)

And means, \* (multiplication), 1 and 0 means 0 because when we see and operator, we multiply the values.

Or means, + (adding), 1 or 0 means 1, because when we see or operator, we add the two values.

Not means, ! (opposite value), Not 1 means 0, because when we use the ! mark in our application it gives opposite value for any boolean variable.

4)

(5 > 4) and (3 == 5) —- False 0

Not (5 > 4 ) —- False 0

(5 > 4 ) or ( 3 == 5) —- True, 1

Not (( 5 > 4) or (3 == 5)) —- False, 0

(True and True) and (True == False) —-- False, 0

(not False) or (not True) —-- True, 1

5)

== =< => < > !=

6)

Assignment operator assigns a value into a variable, while the equal to operator checks if the value is equal to some other value.

7)

Condition is used when a particular values must be met for on a specific line of the code. I would use it when I want to confirm the value of a certain variable while checking it on an already running program.

8)

If spam == 10 - block 1

If spam > 5 - block 2

Else : block 3

9)

Spam = 0

If spam == 1:

print(‘Hello’)

elif(spam == 2):

print(“Howdy”)

Else:

print(“Greetings!”)

10)

CTRL + C

11)

Break will stop the loop, while continue will make the program skip the block and keep running.

12)

range (10) is from 0 to 10

Range (0, 10) is from 0 to 10

Range (0, 10, 1) is still 0 to 10 with 1 increment

They are all same

13)

For i in range(0,11):

print(i)

while (a <= 10):

print(a)

14)

From spam import bacon

bacon()

Extra: round() I mentioned on the first page for Extra, abs() is an absolute number that is calculated when a floating-number is passed into a function.

1)

Functions are useful because we don’t have to write a specific lines of code again and again, we just call a function!

2)

Code runs when the function is called.

3)

Def

4)

Function is defining the doing of function, while the function call is executing it.

5)

Only one global and infinite local scopes.

6)

Nothing

7)

Yes, it can be a part of an expression when passed out to another variable, return value is a result that function could give after running it’s code.

8)

None

9)

Using global keyword

10)

NoneType

11)

It imports areallyyourpetsnamederic library

12)

From spam import bacon

bacon()

13)

Try and catch

14)

Try:

Everything in here will be run once unless there is an exception

Except <Exception name> as e:

print(e)

This will print the specific error and will not cause the program to stop, make it keep running even a n error is occurred.

The Collatz Sequence

def collatz(number):

Try:

if number % 2 == 0:

print(number // 2)

return number // 2

elif number % 2 == 1:

result = 3 \* number + 1 print(result)

return result

Except ValueError as e:

print(“Value must be an Integer”)

print(e)

n = input("Give me a number: ")

while n != 1: n = collatz(int(n))