Columbia Business School
AT THE VERY CENTER OF BUSINESS

Introduction to Python

Session 3

Recall: Data Structures

Dictionary (Dict)

List (List)
Tuple (Tuple)

What you've learned

Boolean (Bool)
String (Str)
Integer (Int)
Float (Float)

Finding the Type of a Variable

```
type(variable_name)
```

Logic Operators

Logic Operators: The Basics

`>`	Greater Than
`<`	Less Than
`==`	Equal To
`!=`	Not Equal To

Logic Operators: The Basics

```
2 < 5
```

```
True
```

```
x = 10
y = 11
x == y
```

```
False
```

```
list1 = [5, 6]
list2 = [5, 6, 7]
list1 != list2
```

```
True
```

Logic operations return Booleans

Logic operations can be used with any data structure

Combining Multiple Logic Operations

and All operations must satisfy at the same time

or At least 1 operation must satisfy

Syntax: (operation1) *and* (operation2) *and* ...

Good coding practice: Use brackets to separate each logic operation

Combining Multiple Logic Operations

```
x = 'hi'
y = 'bye'
z = 'hi'
(x!=y) and (x==z)
```

Note the brackets!

```
True
```

```
x = False
y = False
z = True
(x==y) or (x==z)
```

```
True
```

Logic Operations With Iterables

= Strings, Lists, Tuples, Dictionaries

in Whether an element is inside an iterable

not in Whether an element is NOT inside an iterable

Syntax: element *in* iterable_name

Logic Operations With Iterables

```
tuple_of_letters = ('a', 'b', 'c')
'b' not in tuple_of_letters
```

```
False
```

```
list_of_lists = [[1, 2], [3, 4], [5, 6]]
[3, 4] in list_of_lists
```

```
True
```

Try it!

You are about to go on a Twitter rant, but first, let's check if what you want to say satisfies the 280-character limit.

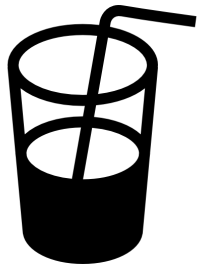
```
rant = 'Despite the constant negative
press covfefe'
```

Your code should return True if your rant satisfies the limit and False if it does not.

```
len(rant) < 280
```


Donald J. Trump
@realDonaldTrump
Despite the constant negative press covfefe
RETWEETS
125,672
LIKES
160,178
12:06 AM - 31 May 2017
41K
126K
160K

Try it!



Your company is in the process of acquiring beverage and clothing companies. Check if either Coca Cola (beverage company) or if H&M (clothing company) has been acquired so far.

```
acquired_beverage_companies = ('Mountain Dew',
'7-Up', 'Pepsi')

acquired_clothing_companies = ('Ralph Lauren',
'H&M')
```

```
('Coca Cola' in acquired_beverage_companies)
or ('H&M' in acquired_clothing_companies)
```

If-Else Statements

If Statements

Any boolean expression
(Recall: A boolean expression is something that evaluates to either True or False)

Syntax

don't forget the colon!

```
if (condition):
    # do something
```

Make sure lines inside the if statement are indented!

- Use a single tab to indent

If Statements

```
if (condition):
    # do something
```

True

```
if 'a' in 'alabama':
    print('Contains "a"')
```

```
Contains "a"
```

If Statements

```
if (condition):
    # do something
```

False

```
if 'z' in 'alabama':
    print('Contains "a"')
```

Nothing is printed

Else Statements

Syntax

```
if (condition):
  # do something
else:
  # do something different
```

don't forget the colons!

Any boolean expression
(Recall: A boolean evaluates to either True or False)

Make sure lines inside the else statement are indented!

Else Statements

```
if (condition):
    # do something
else:
    # do something different
```

True

Else Statements

```
if (condition):
    # do something
else:
    # do something different
```

False

If-Else Statements: Example

Recall: Divides and returns the remainder

```
x = 5
if x % 2 == 0:
    print ('x is even')
else:
    print ('x is odd')
```

```
x is odd
```

Tip: If-Else works well if there are only two outcomes you are interested in

Elif Statements

What if we are interested in more than two outcomes?

Syntax

```
if (condition):
  # do something
elif (condition):
  # do something different
else:
  # do another thing
```

Elif Statements

```
x = 9
if x % 2 == 0:
    print ('x is divisible by 2')
elif x % 3 == 0:
    print('x is divisible by 3')
else:
    print ('x is neither divisible by 2 nor 3')
```

```
x is divisible by 3
```

If-Elif-Else

Tips

- Possible to use *more than one* elif statement for multiple conditions
- Using else statement is optional- think of it as the default option
- Order the statements in priority order- put the most important condition to check first!

Using If vs Elif

```
x = 6
if x % 2 == 0:
    print ('x is divisible by 2')
if x % 3 == 0:
     print('x is divisible by 3')
else:
    print ('x is neither divisible by 2 nor 3')
```

```
x = 6
if x % 2 == 0:
    print ('x is divisible by 2')
elif x % 3 == 0:
     print('x is divisible by 3')
else:
    print ('x is neither divisible by 2 nor 3')
```

Using If vs Elif

```
x = 6
if x % 2 == 0:
    print ('x is divisible by 2')
elif x % 3 == 0:
     print('x is divisible by 3')
else:
    print ('x is neither divisible by 2 nor 3')
```

✓ *Doesn't check the other statements*

```
x is divisible by 2
```

Evaluates only one of the following three statements: if, elif, else

Using If vs Elif

```
x = 6
if x % 2 == 0:
    print ('x is divisible by 2')
if x % 3 == 0:
     print('x is divisible by 3')
else:
    print ('x is neither divisible by 2 nor 3')
```

```
x is divisible by 2
x is divisible by 3
```

Python evaluates the first if statement on its own;
then Python evaluates one of the following two statements: if, else

Try it!

You have just finished lunch with your colleagues. As the server presents the check for your party, you wonder what is the appropriate gratuity given the party size. Write Python code to calculate the total spending including gratuity using the following tipping guidelines:

- If the party has 4 people or less, give a tip of 15% (of the cost of the meal)
- If the party has between 5 and 8 people inclusive, give a tip of 18%
- If the party has more than 8 people, give a tip of 20%

Start with the following example, but write code that would work whatever the value of these variables:

```
party_size = 6

bill = 132
```

Solution

```
party_size = 6
bill = 132
if party_size < 4:
    spending = bill * 1.15
elif 5 <= party_size <= 8:
    spending = bill * 1.18
else:
    spending = bill * 1.20
print(spending)
```

```
155.76
```

Nested If Statements

Syntax

```
if (condition):
  if (condition):
    # do something
  else:
    # do something different
else:
  # do another thing
```

Nested If Statements

```
if (condition):
  if (condition):
    # do something
  else:
    # do something different
else:
  # do another thing
```

The inner code block will be evaluated if the first line evaluates to True

Nested If Statements

```
if (condition):
  if (condition):
    # do something
  else:
    # do something different
else:
  # do another thing
```

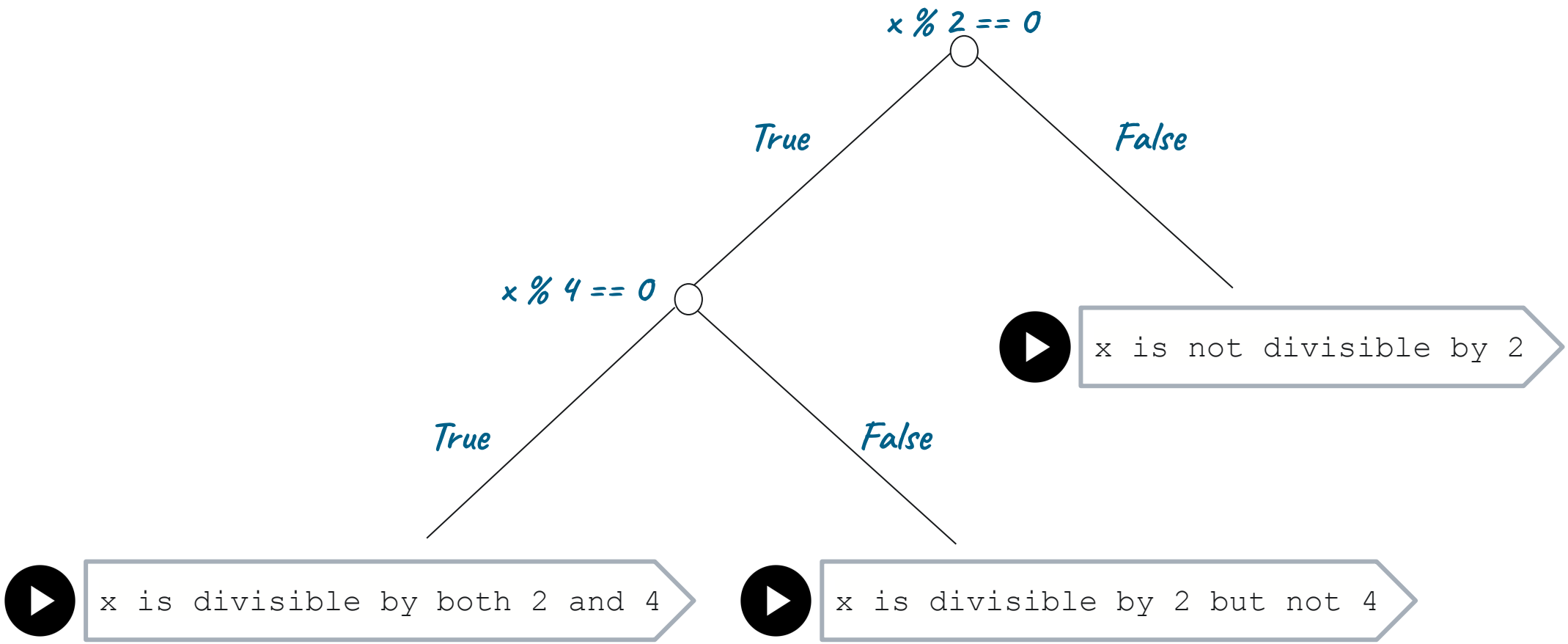
Note the double indentation

Nested If Statements

```
x = 6
if x % 2 == 0:
  if x % 4 == 0:
    print('x is divisible by both 2 and 4')
  else:
    print('x is divisible by 2 but not 4')
else:
  print('x is not divisible by 2')
```

```
x is divisible by 2 but not 4
```

You can think of it as a flow chart



Try it!

You are assessing the valuation of an upscale apartment complex. You want to make adjustments to the valuation determined by your company's proprietary valuation algorithm based on your observations. Based on your experience, you know that a studio apartment and apartments with 1 bedroom or more should be assessed differently.

If the studio apartment has an unblocked view, adjust the valuation by increasing it by 4%. Otherwise, adjust it by decreasing it by 1%. For apartments with bedrooms, adjust its valuation by increasing it by 2% regardless of the view (since these apartments usually have more than one view).

Write Python code to determine the final valuation and use the following Boolean variables: `is_studio, has_view`

Solution

```
if is_studio:
    if has_view:
        final_valuation = valuation * 1.04
    else:
        final_valuation = valuation * 0.99
else:
    final_valuation = valuation * 1.02
print(final_valuation)
```

Loops: While Loops

While Loops

Syntax

```
while (condition):
  # do something
```

Keep repeating loop if condition is true
ie, Keep doing the something as long as condition is still true!

While Loops

Syntax

```
while (condition):
    # do something
```

Make sure all lines in the body of the loop are indented!

While Loops: Example

```
x = 0
list_of_zeros = []
```

Recall: List creation method

```
while x < 3:
  list_of_zeros.append(0)
  x += 1
```

Recall: This is equivalent to `x = x + 1`

```
print(list_of_zeros)
```

Brainstorm: How many times is the while loop repeated?

While Loops: Example

```
x = 0
list_of_zeros = []

while x < 3:
   list_of_zeros.append(0)
   x += 1
```

1st execution of while loop line: `x = 0`

Condition is met → While loop body is executed

After 1st execution of while loop body:

```
print(list_of_zeros)
```

```
[0]
```

```
print(x)
```

```
1
```

While Loops: Example

```
x = 0
list_of_zeros = []

while x < 3:
    list_of_zeros.append(0)
    x += 1
```

2nd execution of while loop line: `x = 1`

Condition is met → While loop body is executed

After 2nd execution of while loop body:

```
print(list_of_zeros)
```

```
[0, 0]
```

```
print(x)
```

```
2
```

While Loops: Example

```
x = 0
list_of_zeros = []

while x < 3:
   list_of_zeros.append(0)
   x += 1
```

3rd execution of while loop line: `x = 2`

Condition is met → While loop body is executed

After 3rd execution of while loop body:

```
print(list_of_zeros)
```

```
[0, 0, 0]
```

```
print(x)
```

```
3
```

While Loops: Example

```
x = 0
list_of_zeros = []

while x < 3:
   list_of_zeros.append(0)
   x += 1
```

4th execution of while loop line: `x = 3`

Condition is NOT met → While loop body is NOT executed

While loop stops with the following:

```
print(list_of_zeros)
```

```
[0, 0, 0]
```

```
print(x)
```

```
3
```

While Loops: Example

```
x = 0
list_of_zeros = []

while x < 3:
  list_of_zeros.append(0)
  x += 1
```

Using while loops is a sleek way of repeating a code *X* number of times- so you don't have to code out the same thing multiple times. Especially useful when *X* is a large number!

Beware: Infinite While Loops!

Infinite Loop

When your code gets stuck inside a loop because the condition never evaluates to `False`

```
while (condition):
    # do something
```

Beware: Infinite While Loops!

What an infinite loop looks like in previous example:

```
x = 0
list_of_zeros = []

while x < 3:
  list_of_zeros.append(0)
```

What changed?

Beware: Infinite While Loops!

What an infinite loop looks like in previous example:

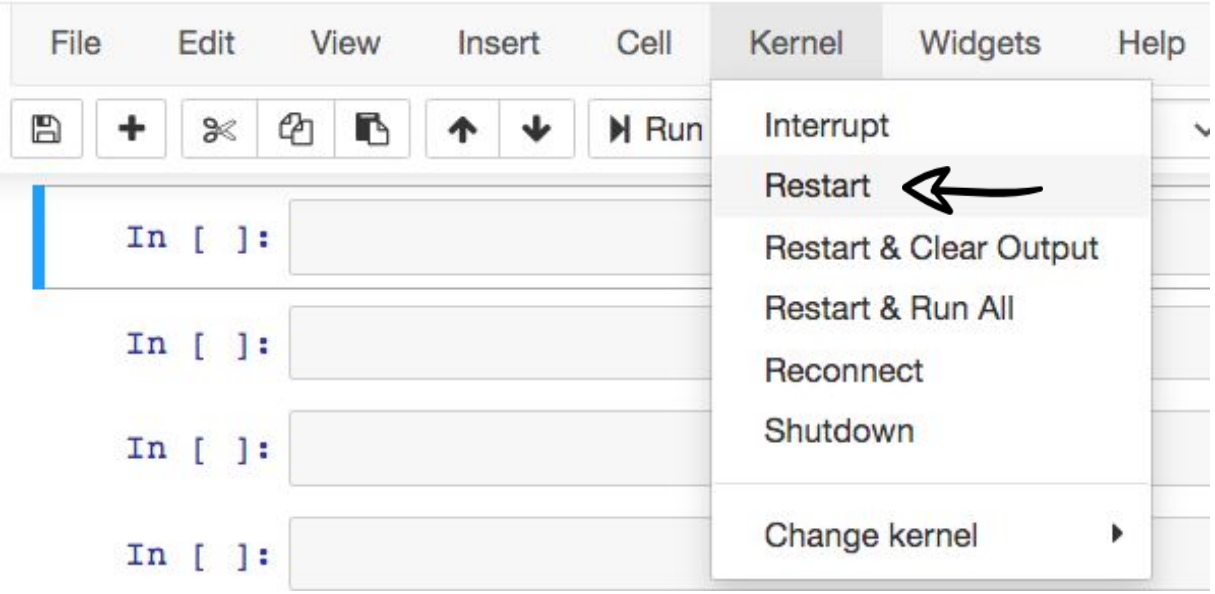
```
x = 0
list_of_zeros = []

while x < 3:
  list_of_zeros.append(0)
  x += 1
```

Without this line, variable *x* always stays at 0

→ Condition (`x < 3`) is always met → While loop repeats infinitely

Beware: Infinite While Loops!

How to get out of it? On Jupyter Notebook: Restart your kernel



Note: Restarting a notebook kernel means all your previously created variables will be lost from memory, so you'll have to re-run previous code in order to recreate those variables, if you need to use them in subsequent code

While Loops: Concepts

- **All lines** inside the loop body will run if the condition is satisfied. They are run one at a time and in order of the code written

- The while loop condition is checked again only after **all lines inside its body** have been run

- It’s possible for a while loop body to not be run at all, if the condition is never satisfied

- Beware of infinite loops!

Try it!

You are given the following variable *x*:

```
x = 3
```

Use a while loop to successively increase the value of the variable x by 2 until it becomes bigger than 10.

Solution

```
while x < 10:
  x += 2
```

While loop body stops executing when the value of x reaches 11

Try it!

You are looking to make investments. You have a list of possible investments:

```
possible_investments = [1922, 352, 2454, 3565]
```

Find out the number of investments you can make given a budget of $3000, assuming you must invest from left to right of the above list.

```
number = 0
sum = 0

while sum < 3000:
  sum += possible_investments[number]
  number += 1

print('Can make {number+1} investments.')
```

Recall: Printing with f-strings

Because *number* started at 0!

Loops: For Loops

For Loops

Syntax

```
for elem in iterable_name:
  # do something
```

= Strings, Lists, Tuples, Dictionaries

Provides a faster way to loop through a sequence and iterate (perform something repeatedly) than while loops

For Loops

Syntax

```
for elem in iterable_name:
    # do something
```

Make sure all lines in the body of the loop are indented!

For Loops: List Example

```
states = ['NY', 'CA', 'FL', 'MA']
for state in states:
    print(state)
```

Goes across the list sequentially, starting from 'NY' and ending at 'MA'
Repeats the loop body once for each element in the list

refers to each element in the iterable
(can be named anything)

```
NY
CA
FL
MA
```

For Loops: String Example

Recall: Strings are also iterables and can be iterated over!

```
states = 'Texas'
for letter in states:
    print(letter)
```

Goes across the string sequentially, starting from 'T' and ending at 's'
Repeats the loop body once for each element (character) in the string

refers to each element in the iterable (can be named anything)

```
T
e
x
a
s
```

Let’s see how for loops can be much faster

Suppose we want to correct a clerical error and convert all elements in a list to be uppercase.

```
stocks = ['fb', 'aapl', 'nflx', 'goog']
print(stocks[0].upper())
print(stocks[1].upper())
print(stocks[2].upper())
print(stocks[3].upper())
```

a lot of repeated code!

```
FB
AAPL
NFLX
GOOG
```

A much better approach

```
stocks = ['fb', 'aapl', 'nflx', 'goog']

for stock in stocks:
    print(stock.upper())
```

Goes across the list sequentially, starting from 'fb' and ending at 'goog'
Repeats the loop body once for each element in the list
= Converts each element into uppercase iteratively

```
FB
AAPL
NFLX
GOOG
```

Try it!

You are about to send out a monthly email to your clients to update them about your company's latest products. Your clients' emails are:

```
list_of_emails = ["ej9212@columbia.edu",
"sj4837@harvard.edu", "jk6666nyu.edu"]
```

Write code that checks if the database of emails you have is valid, ie. check if each email address contains the symbol '@'.

Solution

```
for email in list_of_emails:
    print("@" in email)
```

Running code a fixed number of times

Syntax

```
for i in range(number):
  # do something
```

range(*number*) returns a sequence of numbers that starts from 0 and increments by 1 (by default), until it reaches *number-1*

```
x = range(3, 6)
for num in x:
    print(num)
```

```
3
4
5
```

Running code a fixed number of times

Syntax

```
for i in range(number):
  # do something
```

Using range(*n*) runs the for loop body *n* times!

For Range: Example

```
employees = 5
expenditure = 0

for i in range(employees):
    expenditure += 1500
```

For loop body is repeated 5 times

```
print(expenditure)
```

```
7500
```

1500 is added 5 times to 0

Break

`break` stops the entire for loop execution
= break out of the for loop code block

```
stocks = ['fb', 'aapl', 'nflx', 'goog']
for stock in stocks:
    if stock == 'aapl':
        break
    else:
        print(stock.upper())
```

```
FB
```

Continue

`continue` stops the current loop body execution and continues with the loop body execution using the next element in the sequence

```
stocks = ['fb', 'aapl', 'nflx', 'goog']
for stock in stocks:
    if stock == 'aapl':
        continue
    else:
        print(stock.upper())
```

```
FB
NFLX
GOOG
```

Try it!

You are keeping track of your weekly budget in a list.
Write code to calculate your average daily expenditure.

```
budget = [42, 102, 12, 63, 9, 88, 65]
```

Be sure to check that your code works for lists of different lengths.

Solution

```
total = 0
for expenses in budget:
    total += expenses
avg = total/len(budget)
print(avg)
```

```
54.42857142857143
```

Solution: Extra

Two ways of rounding the values to a certain decimal point:

- Using f-string
- Using round(*value, decimal point to be rounded to*)

```
print(f"{avg:.2f}")
print(round(avg, 2))
```

```
54.43
54.43
```

Try it! The Fizz Buzz Challenge

Write a program that looks at numbers from 1 to 100. For multiples of three, print "Fizz" instead of the number and for the multiples of five, print "Buzz". For numbers that are multiples of both three and five print "FizzBuzz". For all other numbers, simply print the number.

Try it! The Fizz Buzz Challenge

Fizz Buzz Test

The "Fizz-Buzz test" is an interview question designed to help filter out the 99.5% of programming job candidates who can't seem to program their way out of a wet paper bag. The text of the programming assignment is as follows:

https://wiki.c2.com/?FizzBuzzTest

Will this get the desired result?

```
for number in range(1, 101):
    if (number % 3) == 0:
        print("Fizz")
    elif (number % 5) == 0:
        print("Buzz")
    elif ((number % 3) == 0) and ((number % 5) == 0):
        print("FizzBuzz")
    else:
        print(number)
```

Solution

```
for number in range(1, 101):
    if ((number % 3) == 0) and ((number % 5) == 0):
        print("FizzBuzz")
    elif (number % 3) == 0:
        print("Fizz")
    elif (number % 5) == 0:
        print("Buzz")
    else:
        print(number)
```