Python Fundamentals, Data Structures, and Algorithms:

Week 1 Workshop Presentation

Your Instructor

Jack Seymour

Bellingham Washington

Instructor
Entrepreneur
Security Researcher

I own and operate Collected Worlds.
I was on the team that shipped the Xbox.
I have been working in the industry for more than 20 years.

Workshop Agenda

Activity	Estimated Duration
Introductions & Ice Breakers	30 mins
Your Bootcamp Checklist	20 mins
Review	60 mins
Break	15 mins
Workshop Assignment	100 mins
Check-Out (Feedback & Wrap-Up)	15 mins

Student introductions

1. Name, or preferred nickname.
2. Where did you grow up? Where do you live now?
3. Self-identified comfort level in Python, SQL, or DevOps? (somewhat comfortable, comfortable, very comfortable…?)
4. What motivates you to attend this bootcamp?
5. Share one “fun fact” about yourself. Or, share one of your answers from the Icebreaker (best/worst job, moment you would reply, etc).

What to expect every week

Next week's content is unlocked after this week's workshop, once you have completed the Feedback

Daily tasks **every day (2+ hours)**:

- ✓ Work through lessons, exercises, and challenges, in order of appearance
- ✓ Additional exploration online
- ✓ Participate in Slack
- ✓ Work on Portfolio Project

End-of-week Quiz

4-hour workshop every Saturday

- ✓ Read the Workshop Assignment instructions ahead of time

IT WILL BE HARD
IT WILL IMPACT YOUR ROUTINE

The 20 minutes rule

Try for 20 minutes, then ask for help!

If you ask for help too soon:

You will not learn how to tackle problems or remove obstacles, which is crucial for coding.
Understanding the process or path that resolved the issue is more important than the solution.

If you ask for help too late:

You will get frustrated and tired.
You will miss an opportunity to go deeper on the same topic. Time is limited!

10 minutes rule during workshops

During the week, go by the 20 minutes rule.
During workshops, go by a 10 minutes rule – try to solve the issue yourself (or with your classmate, if working together) for 10 minutes before asking your instructor for help.

Bootcamp Checklist

- ✓ You receive daily emails
- ✓ Code of Conduct
- ✓ Installed VS Code
- ✓ Installed Moodle app on phone
- ✓ Set up NucampFolder
- ✓ Installed Developer Tools
- ✓ Learned Bash CLI use
- ✓ Installed Python
- ✓ Installed Python Tools

Your time commitment pledge

- **Time Commitment Pledge:**
  - 14 to 20 hours of focused time every week.
- **Weekly Commitment Pledge:**
  - Attend every Saturday workshop.
  - Spend at least two or more hours every day on the course, whether via the course content, working on your Portfolio Project, or exploring additional materials online.
  - Publish your workshop assignment on Saturday or Sunday (**avoid late submissions**).
- **Talk to your Friends and Relatives:**
  - Have a conversation with your family to explain to them your commitment pledge and seek their support.
  - Similarly, reach out to your friends and seek their support.
  - Don't underestimate the importance of communicating your intentions to those around you!

Week 1 Review

Week 1 Review

Overview

Python version check	Python indentation
Data types	Code blocks
Arithmetic operators	While loops
Operator precedence	Break
Comparison operators	Continue
Logical operators	Infinite while loops
Conditional branching	Input function

Review: Installation

- Confirm: Does everyone have the correct version of python installed? Version: 3.9.4

- Open VS Code’s integrated terminal and enter the command:

```
python -V
```

Review: Primitive data types

Python offers several data types for storing and managing information.

What have we learned about the 4 main primitive (basic) data types?

- int
- float
- string
- boolean

Review: Composite data types

- Python also offers composite data types
- Composite data types can be a combination of more than one data type

The 4 primary composite data types:

- list
- dict
- tuple
- set

Review: Arithmetic operators

+	:	Addition
-	:	Subtraction
*	:	Multiplication
/	:	Division
%	:	Modulo
**	:	Exponentiation
//	:	Floor Division

Review: Arithmetic operators

What is the result of...

1) 2 ** 3
2) 20 % 4
3) 23 % 4
4) 23 // 4

Review: Arithmetic operators

Answers:

1) 2 ** 3	8
2) 20 % 4	0
3) 23 % 4	3
4) 23 // 4	5

Review: Operator precedence

Python uses operator precedence,
which you may recall from math class in school

Precedence	Operator(s)	Operation(s)
1	( )	Parentheses
2	**	Exponention
3	* / %	Multiplication Modulo Division
4	+ -	Addition Subtraction

NOTE: Operators with same precedence
must be applied in order from left to right
You can use the acronym **PEMDAS** to help remember the order.

Review: Operator precedence

Compute the values of the following:

x = 2 * (3 + 4**2) / 19

y = 1 + 2 ** 3 / 4 – 5 * 6

z = 15 / 3 * (2 * 6 / 3)

Review: Operator precedence

Answers

x = 2

y = -27

z = 20

Review: Comparison operators

Comparison operators are typically used to control the flow of your program, e.g.:
If x < y then do task A otherwise do task B

Comparison Operators	Boolean Expression	Meaning
>	X > Y	X is greater than Y
<	X < Y	X is less than Y
>=	X >= Y	X is greater than or equal to Y
<=	X <= Y	X is less than or equal to Y
==	X == Y	X is equal to Y
!=	X != Y	X is not equal to Y

Review: Comparison operators

X = 3
Y = 2

Discuss:

1) What is the value of **X <= Y** ?
2) What is the value of **X == Y** ?
3) What is the value of **X != Y** ?

Review: Comparison operators

X = 3
Y = 2

Discuss:

1) What is the value of **X <= Y** ? Boolean False
2) What is the value of **X == Y** ? Boolean False
3) What is the value of **X != Y** ? Boolean True

Review: Logical operators

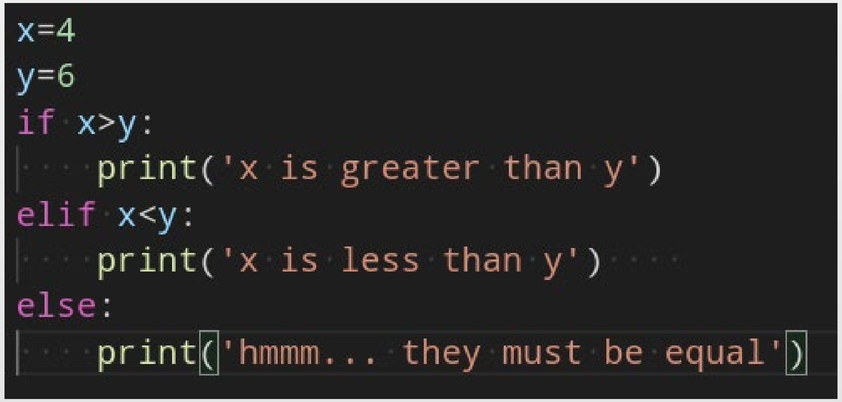
Logical operators **and**, **or**, and **not** are used to join two or more expressions to create more complex conditions.

Lotto-Mania

Day	**Number**	**Age**	**Day == “Mon” and Num == 749 and Age >= 18**	**Day == “Sat” and Num == 145 and Age >= 18**
Mon	749	12	Lose	Lose
Wed	86	18	Lose	Lose
Sat	145	28	Lose	Win

Day	**Number**	**Age**	**Day == “Mon” or Num == 86 or Age >= 18**	**Day == “Fri” or Num == 56 or not Age >= 18**
Mon	749	12	Win	Win
Wed	86	18	Win	Lose
Sat	145	28	Win	Lose

Review: Conditional branching



```
x=4
y=6
if x>y:
    print('x is greater than y')
elif x<y:
    print('x is less than y')
else:
    print('hmmm... they must be equal')
```

The **if** statement is used for conditional branching

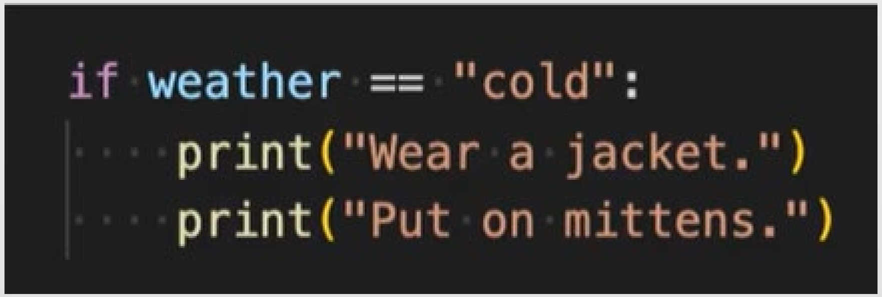
If the condition has been met (is true) then all code that should be executed under that condition must be indented together

Review: Python indentation

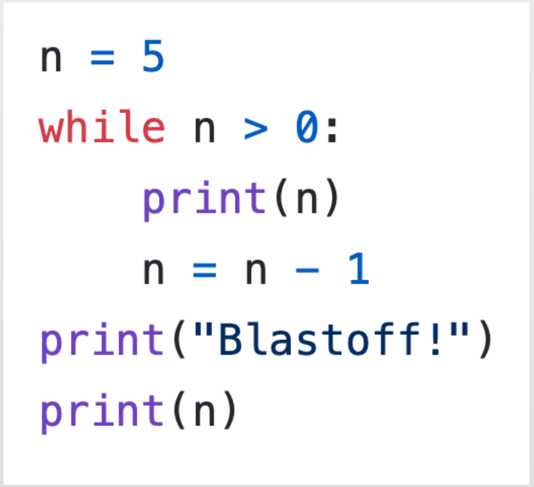
- General standard: 4 spaces per indentation level
- You can use spaces or tabs
- Indentation style must be consistent within same file
- Otherwise, Python will error

Review: Code blocks

- Code indented at the same level creates a code block
- Example:



```
if weather == "cold":
    print("Wear a jacket.")
    print("Put on mittens.")
```



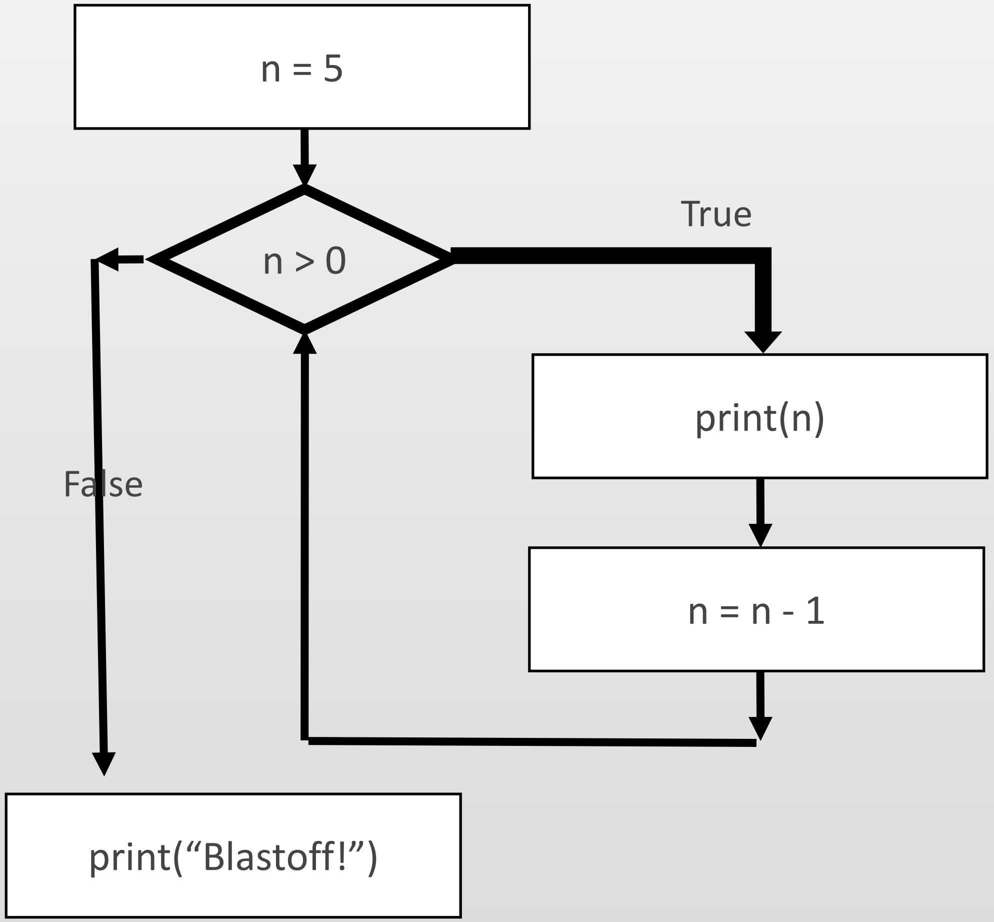
```
n = 5
while n > 0:
    print(n)
    n = n - 1
print("Blastoff!")
print(n)
```

Review: While loops

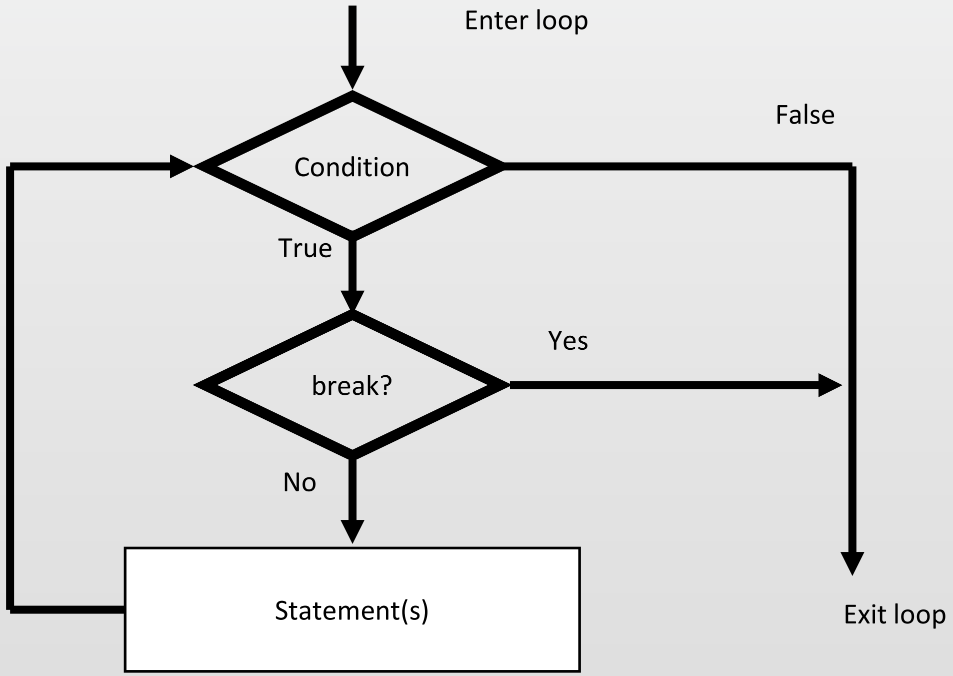
```python
n = 5
while n > 0:
    print(n)
    n = n - 1
print("Blastoff!")
print(n)
```

Output:

```
5
4
3
2
1
Blastoff!
0
```

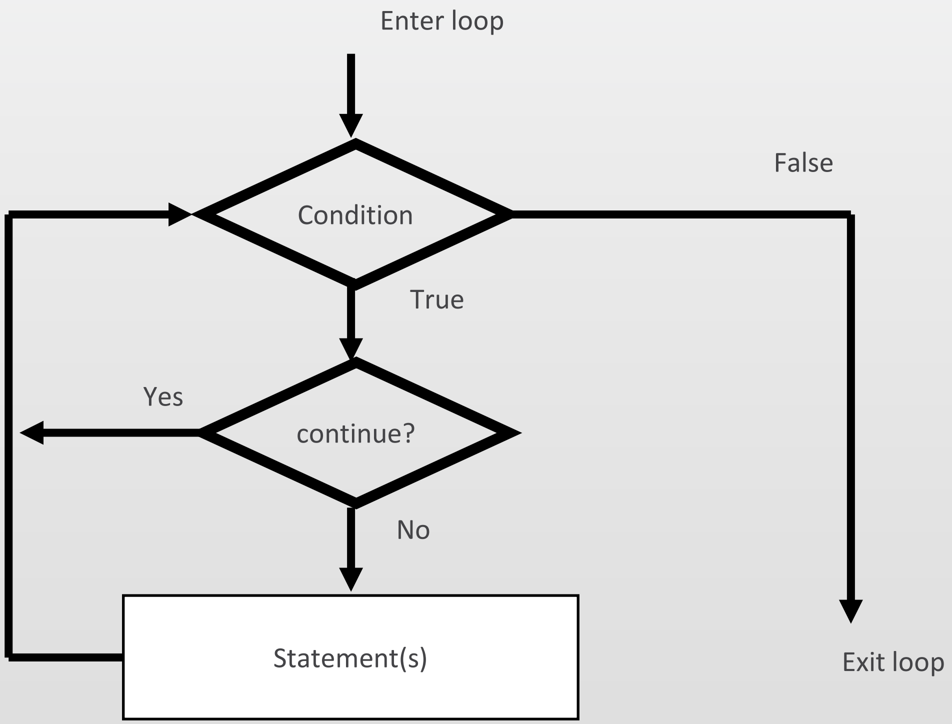


Review: Loop control - break



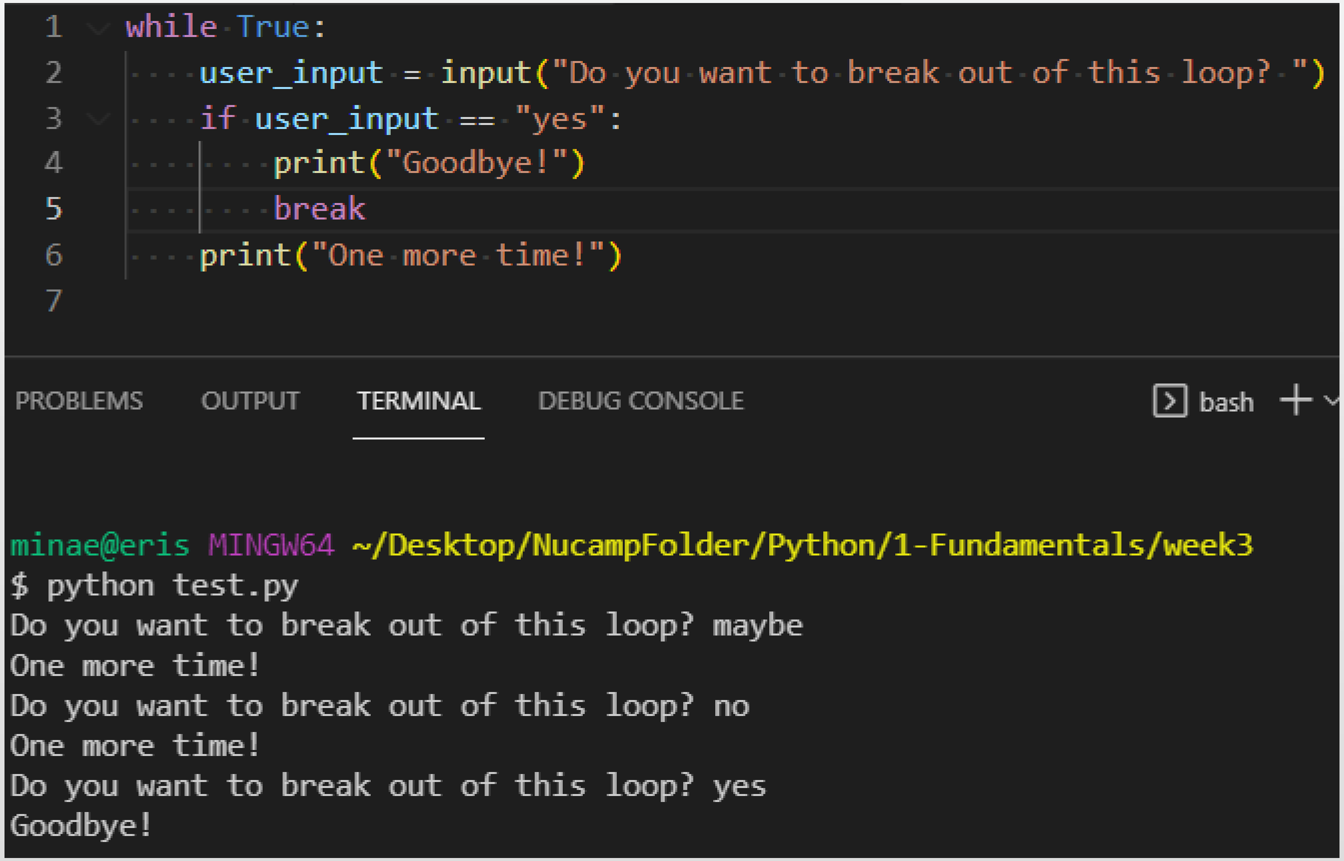
```
n = 5
while n > 1:
    print(n)
    n = n - 1
    if n == 2:
        break
print(n)
# output: 5 4 3 2
```

Review: Loop control - continue



```
print("h")
n = 5
while n > 1:
    n = n - 1
    if n == 2:
        continue
    print(n)
# output: 4 3 1
```

Review: Infinite while loops

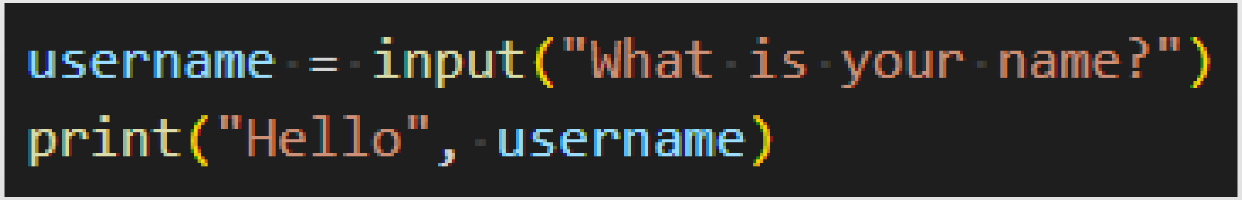


```python
while True:
    user_input = input("Do you want to break out of this loop? ")
    if user_input == "yes":
        print("Goodbye!")
        break
    print("One more time!")
```

```
minae@eris MINGW64 ~/Desktop/NucampFolder/Python/1-Fundamentals/week3
$ python test.py
Do you want to break out of this loop? maybe
One more time!
Do you want to break out of this loop? no
One more time!
Do you want to break out of this loop? yes
Goodbye!
```

Review: Input function

- The **input** function is used in Python to retrieve keyboard input from the user
- As a function, **input()** must be called using parentheses
- Use a string argument to prompt the user what to input.
- The return value of the **input()** function is a string
  - You may need to convert the value to a different data type



```python
username = input("What is your name?")
print("Hello", username)
```

Workshop 1 Assignment

Goal: Code a fantasy battle game!

Task 1

Set up your game variables: the game characters and their stats.

Task 2

Prompt the player to choose from a list of options.

Task 3

Set up infinite while loop to handle player choice.

Task 4

Battle with the Dragon!

You will be split up into groups to work on the assignment together.
Talk through each step out loud with each other, code collaboratively.
If your team spends more than 10 minutes trying to solve one problem,
ask your instructor for help!