Hands-on Python Foundations

Day 1:

Writing simple scripts

Outline

Today's topics

- Running Python – interactive and script mode
- Math
- Variables
- Types
- Text formatting
- Functions
- Boolean expressions
- Conditionals
- Loops

Interactive Scenarios – Day 1

1. Getting started

- https://learning.oreilly.com/scenarios/hands-on-python-foundations/9780137904648X001/

2. Types, variables and strings

- https://learning.oreilly.com/scenarios/hands-on-python-foundations/9780137904648X002/

3. Functions and control flow

- https://learning.oreilly.com/scenarios/hands-on-python-foundations/9780137904648X003/

Schedule

0:00 - Intro and setup

0:40 - **Getting started with Python** - Interactive scenario #1

1:20 - *Break* (15 mins)

1:35 - **Python basics – types, variables and strings** - Interactive scenario #2

2:35 - *Break* (15 mins)

2:50 - **Python scripts – Functions, conditionals and loops** - Interactive scenario #3

3:50 - Intro to homework (5 min)

3:55 - Quiz (5 min)

Questions and breaks

- I'll answer attendee chat throughout class
- Q&A widget during the breaks

- 2 Breaks (15 mins each)

- Email more in-depth questions at arianne.dee.studios@gmail.com

Setup your environment

1. Install Python
2. Install PyCharm Community
3. Install Git
4. Clone course content
5. Make sure PyCharm is configured

Instructions - https://github.com/ariannedee/python-foundations-3-weeks/

Introductions

Poll #1

- How much Python do you know?
  - Absolutely none
  - Almost nothing
  - Some
  - A fair amount
  - A lot

Poll #2

- How much programming do you know?
  - Absolutely none
  - Almost nothing
  - Some
  - A fair amount
  - A lot

Poll #3 (multi-choice)

- Why are you taking this course?
  - Curiosity / for fun
  - Potential career change
  - Improve technical knowledge
  - Incorporate it into work (as a non-developer)
  - Use it at work (as a developer)
  - For my hobby/side-project

Poll #4 (multi-choice)

- What applications are you hoping to use it for?
  - Data science / machine learning
  - Dev ops / server management
  - Web applications
  - Desktop applications
  - Raspberry Pi / IoT
  - Scripting / web scraping
  - Mobile development
  - Game development
  - other

About me

- UBC Civil Engineering (2009) and Computer Science (2014)

- Past employers:
  - ThoughtWorks (Java)
  - Sensible Building Science (lead/sole Django dev)
  - 7Geese (full-stack Django + React)
  - Freelance/contractor since 2017








O'Reilly Live Trainings

- Introduction to Python Programming
- Programming with Python: Beyond the Basics
- Python Environments and Best Practices

- Hands-On Python Foundations in 3 Weeks
  - Incorporates content from the 3 classes above

- Object-Oriented Programming in Python
- Introduction to Django: a web application framework for Python
- Rethinking REST: A hands-on guide to GraphQL and queryable APIs
- See all

O'Reilly Videos

- Introduction to Python
- Next Level Python
- Rethinking REST: A hands-on guide to GraphQL and Queryable APIs

- See all

- Links to relevant video lessons in repository README
  - https://github.com/ariannedee/python-foundations-3-weeks

Course Project

Automated daily email

Good morning, Arianne 👋

Today's weather ☁️

Today there will be light snow.
High: 5.9 °C (42.6 °F)
Low: -0.9 °C (30.3 °F)

☀️ 7:48 AM
🌙 5:16 PM

Joke of the day 🤣

Little Johnny comes home from his first day of school. His mother asks, "So, what did you learn at school today?" Little Johnny replies, "NOT ENOUGH. They want me to come back tomorrow!"

Quote of the day 💬

When you recover or discover something that nourishes your soul and brings joy, care enough about yourself to make room for it in your life.
~ Jean Shinoda Bolen

Next game 🏀

In 2 days @ 7:30 PM ET
Home: Toronto Raptors
Away: Atlanta Hawks

Project

- Find some free APIs to retrieve data from
  - weather
  - quote/joke of the day
  - calendar, reminders, todos
  - sports, stocks, events
  - Find 2-3 options

- Send email every morning

- Basic – plain text email
- Advanced – HTML (formatted) email

Set up

Setup your environment

1. Install Python
2. Install PyCharm Community
3. Install Git
4. Clone course content
5. Make sure PyCharm is configured

Instructions - https://github.com/ariannedee/python-foundations-3-weeks/

Windows links

1. Install Python
   - https://www.python.org/downloads/
   - IMPORTANT instructions
2. Install PyCharm Community
   - https://www.jetbrains.com/pycharm/download/
3. Install Git
   - https://git-scm.com/download/win
4. Clone course content
   - git@github.com:ariannedee/python-foundations-3-weeks.git
5. Make sure PyCharm is configured

Mac/Linux links

1. Install Python
   - https://www.python.org/downloads/
2. Install PyCharm Community
   - https://www.jetbrains.com/pycharm/download/
3. Install Git
   - https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
4. Clone course content
   - git@github.com:ariannedee/python-foundations-3-weeks.git
5. Make sure PyCharm is configured

Today's topics

- ❑ Running Python – interactive and script mode
- ❑ Math
- ❑ Variables
- ❑ Types
- ❑ Text formatting
- ❑ Functions
- ❑ Boolean expressions
- ❑ Conditionals
- ❑ Loops

Today's topics

- ❑ **Running Python – interactive and script mode** ←
- ❑ Math
- ❑ Variables
- ❑ Types
- ❑ Text formatting
- ❑ Functions
- ❑ Boolean expressions
- ❑ Conditionals
- ❑ Loops

Running Python

Interactive Scenario # 1

1. Getting started

- https://learning.oreilly.com/scenarios/hands-on-python-foundations/9780137904648X001/

Running Python

- **Interactive mode**
  - Run line-by-line
  - Good for testing stuff and noodling around

- **Script mode**
  - Run a file
  - Build scripts, programs and applications
  - Run the same thing multiple times

Applications to use

- **IDLE**
  - Comes with Python install

- **Terminal/Powershell**
  - Comes with operating system

- **IDE – PyCharm, VSCode, etc.**
  - Download from website

- **Jupyter Lab**
  - Install with pip - instructions

PyCharm

- https://www.jetbrains.com/pycharm/download/
- Community edition is free to use
- Built specifically for Python
  - Syntax and error highlighting
  - Refactoring features
  - Code completion, navigation, documentation

Open the code in PyCharm

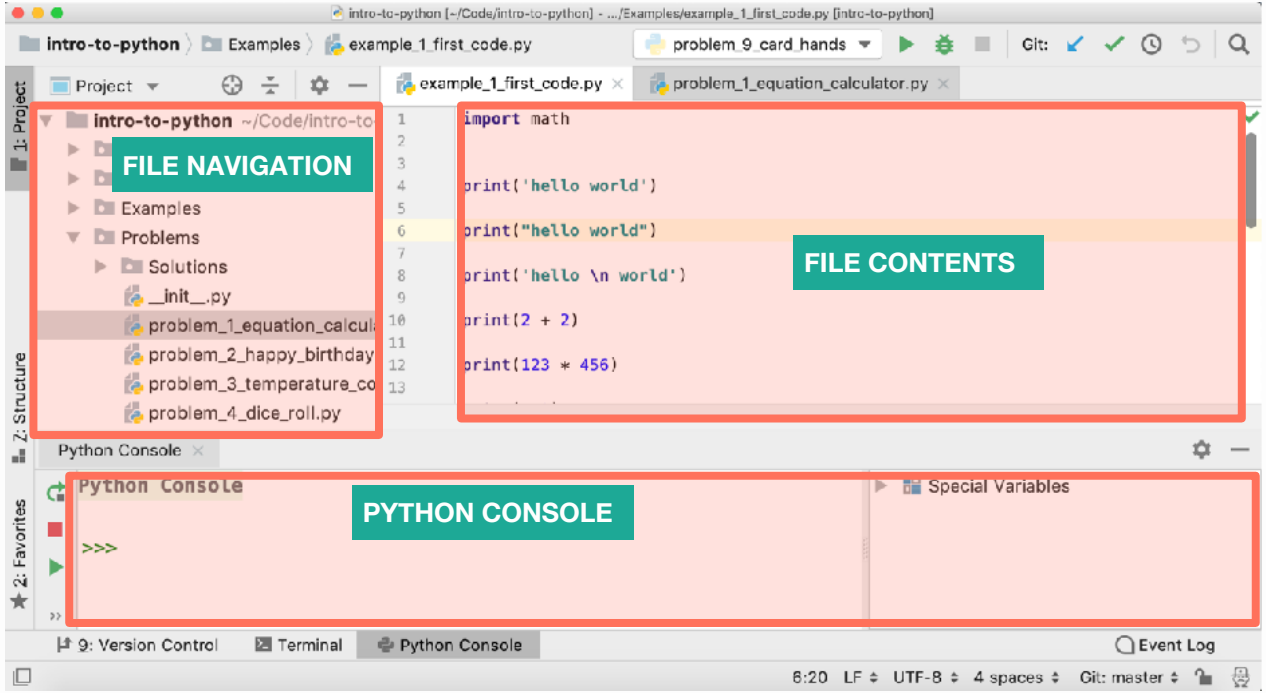
- Clone the code in PyCharm
  - See instructions

Or

- Clone the code on the command line
- Open the *python-foundations-3-weeks* folder in PyCharm

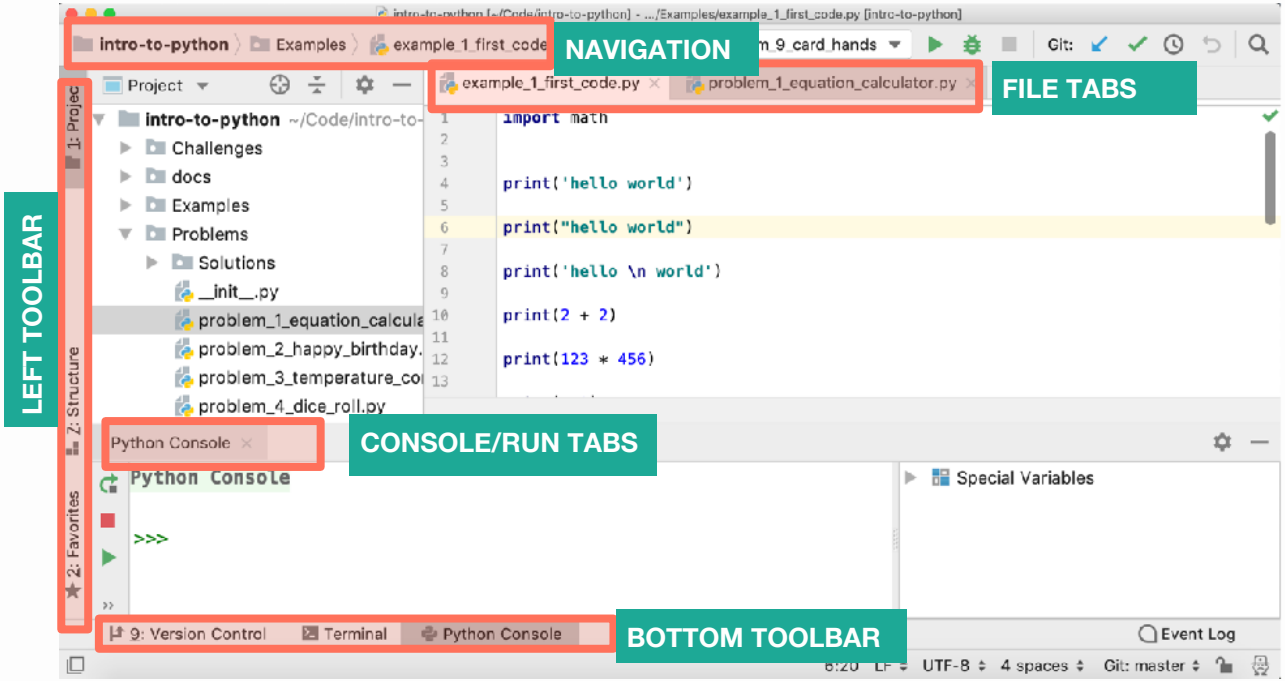
PyCharm overview

PyCharm Layout



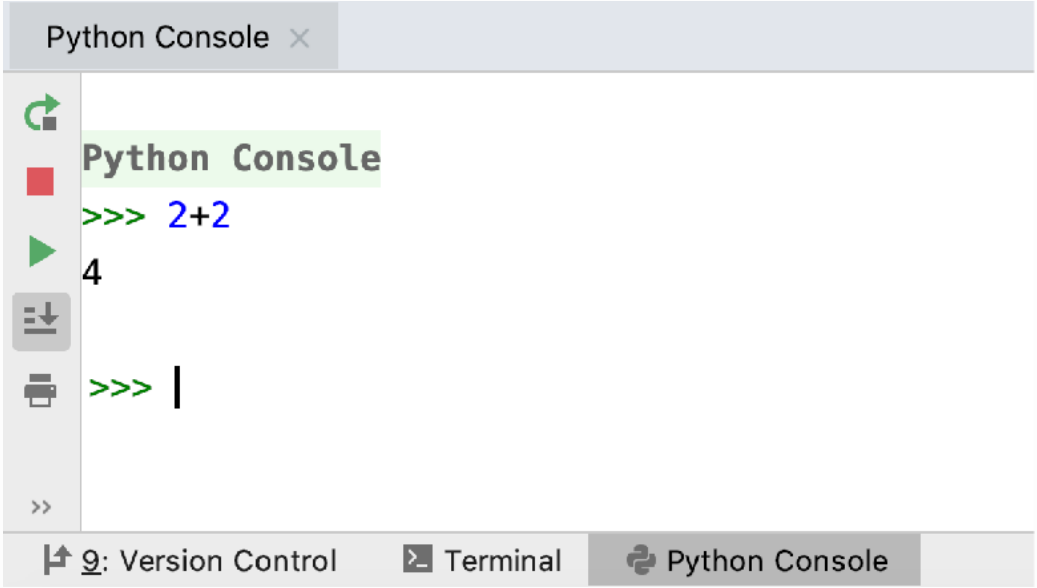
Reference: page 2

PyCharm Toolbars



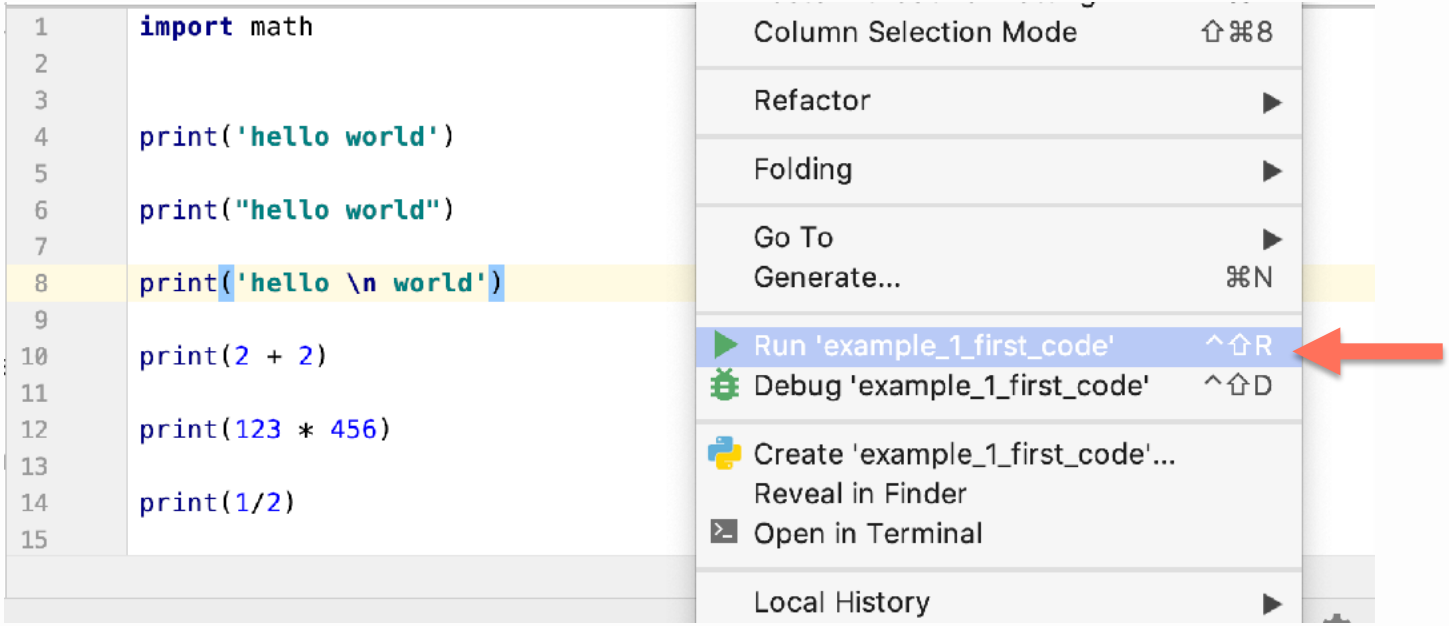
Reference: page 2

Run code in the console



Run code from a file

Right click in file



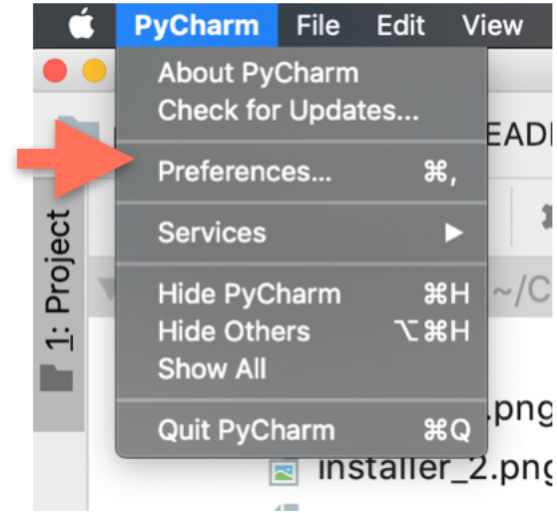
If PyCharm doesn’t recognize Python3

Reference PDF: page 4-6

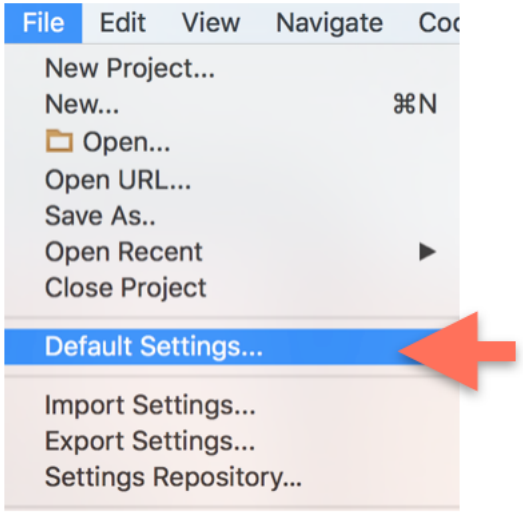
GitHub course documentation: link

Settings

Mac



PC



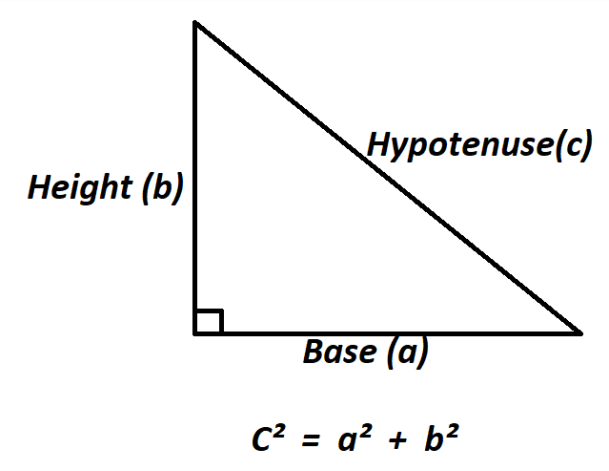
Follow along

- Katacoda
- PyCharm
- VS Code

Today's topics

- ☑ Running Python – interactive and script mode
- ☐ **Math** 
- ☐ Variables
- ☐ Types
- ☐ Text formatting
- ☐ Functions
- ☐ Boolean expressions
- ☐ Conditionals
- ☐ Loops

problem_1_hypotenuse.py



$$c = \sqrt{(a^2 + b^2)}$$

About Python

“Hello World” in different languages

Roughly from high - low level of abstraction

“Hello World” in different languages

Python

```
print("Hello World")
```

“Hello World” in different languages

JavaScript

```
console.log("Hello World!");
```

“Hello World” in different languages

Java

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Prints the string to the
console.
    }
}
```

“Hello World” in different languages

C++

```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";
    return 0;
}
```

“Hello World” in different languages

Assembly

```
    global  _main
    extern  _printf

    section .text
_main:
    push    message
    call    _printf
    add     esp, 4
    ret
message:
    db  'Hello, World', 10, 0
```

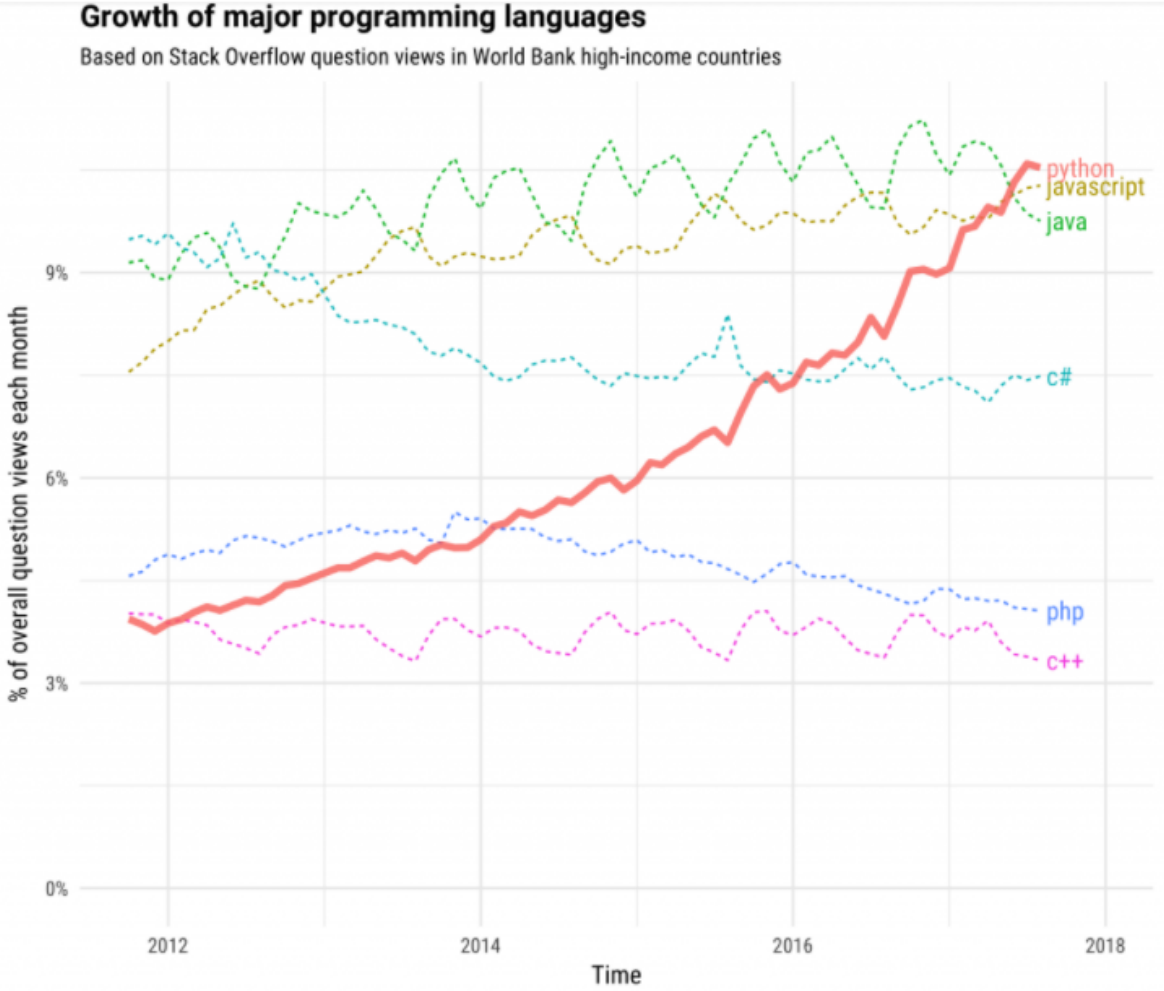
“Hello World” in different languages

Machine Code

```
b8    21 0a 00 00   #moving "!\n" into eax
a3    0c 10 00 06   #moving eax into first memory location
b8    6f 72 6c 64   #moving "orld" into eax
a3    08 10 00 06   #moving eax into next memory location
b8    6f 2c 20 57   #moving "o, W" into eax
a3    04 10 00 06   #moving eax into next memory location
b8    48 65 6c 6c   #moving "Hell" into eax
a3    00 10 00 06   #moving eax into next memory location
b9    00 10 00 06   #moving pointer to start of memory location into
ecx
ba    10 00 00 00   #moving string size into edx
bb    01 00 00 00   #moving "stdout" number to ebx
b8    04 00 00 00   #moving "print out" syscall number to eax
cd    80            #calling the linux kernel to execute our print to
stdout
b8    01 00 00 00   #moving "sys_exit" call number to eax
cd    80            #executing it via linux sys_call
```

Python

- High-level language
  - Is closer to English than most others
- Simple syntax
  - Easy to learn and get stuff done
- Open source
  - Everything is free, lots of things are well-maintained



Great for

- Prototyping
- Scripting (automation tasks, managing servers)
- Data analysis and machine learning
- Teaching
- Low - medium traffic web apps
- RaspberryPi

Not great for

- High speed applications
- Multi-threaded applications
- Mobile development
- Easy to learn, hard to master and progress

Where is it used?

Web apps

Data analysis

More common options

- **Desktop apps**
  - Java, Swift/Objective-C *(Mac)*, C# *(Windows)*, JavaScript *(with Electron)*
- **Mobile apps**
  - Kotlin/Java*(Android)*, Swift/Objective-C *(iOS)*, C# *(with Unity)*, JavaScript *(with React Native)*
- **High speed, high reliability, multi-threading**
  - C/C++, Go, Rust

About Python

About Python

- Released in 1990
- Created by Guido Van Rossum
- Python Enhancement Proposals (PEPs)

About Python

- Released in 1990
- Created by Guido Van Rossum
- Python Enhancement Proposals (PEPs)

About Python

- Released in 1990
- Created by Guido Van Rossum
- Python Enhancement Proposals (PEPs)

Style Guide (PEP 8)

Indentation

Use 4 spaces per indentation level.

Tabs or Spaces?

Spaces are the preferred indentation method.

Tabs should be used solely to remain consistent with code that is already indented with tabs.

Zen of Python (PEP 20)

Beautiful is better than ugly
Explicit is better than implicit
Simple is better than complex
Complex is better than complicated
Readability counts
…

- Try typing "`import this`" into the interpreter
- Next, try "`import antigravity`"

“Code is more often read than written.”

- Guido van Rossum

More about Python

- Why you should learn Python
  - https://yourstory.com/mystory/interesting-facts-about-python-language
- Python Developer Survey 2019
  - https://www.jetbrains.com/lp/python-developers-survey-2019/
- StackOverflow developer survey 2019
  - https://insights.stackoverflow.com/survey/2019#technology
- Python fun facts
  - https://data-flair.training/blogs/facts-about-python-programming/

Q&A and 15 min break

Interactive Scenario #2

2. Types, variables and strings

- https://learning.oreilly.com/scenarios/hands-on-python-foundations/9780137904648X002/

Today's topics

- ☑ Running Python – interactive and script mode
- ☑ Math
- ☐ **Variables** ←
- ☐ Types
- ☐ Text formatting
- ☐ Functions
- ☐ Boolean expressions
- ☐ Conditionals
- ☐ Loops

Python keywords

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Covered keywords – week 1

False	await	**else**	**import**	pass
None	**break**	except	**in**	raise
True	class	finally	**is**	**return**
and	**continue**	**for**	lambda	try
as	**def**	**from**	nonlocal	**while**
assert	**del**	global	**not**	with
async	**elif**	**if**	**or**	yield

Variables

False	await	**else**	**import**	pass
None	**break**	except	**in**	raise
True	class	finally	**is**	**return**
and	**continue**	**for**	lambda	try
as	**def**	**from**	nonlocal	**while**
assert	**del**	global	**not**	with
async	**elif**	**if**	**or**	yield

Today's topics

- ☑ Running Python – interactive and script mode
- ☑ Math
- ☑ Variables
- ☐ **Types** ←
- ☐ Text formatting
- ☐ Functions
- ☐ Boolean expressions
- ☐ Conditionals
- ☐ Loops

Types

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Today's topics

- ☑ Running Python – interactive and script mode
- ☑ Math
- ☑ Variables
- ☑ Types
- ❑ **Text formatting** ←
- ❑ Functions
- ❑ Boolean expressions
- ❑ Conditionals
- ❑ Loops

problem_2_greeting.py

Good morning, **{name}**!
Today is going to be **{condition}**.
High: **{high_c}** °C (**{high_f}** °F)
Low: **{low_c}** °C (**{low_f}** °F)

15 min break

Interactive Scenario #3

3. Functions and control flow

- https://learning.oreilly.com/scenarios/hands-on-python-foundations/9780137904648X003/

Today's topics

- ☑ Running Python – interactive and script mode
- ☑ Math
- ☑ Variables
- ☑ Types
- ☑ Text formatting
- ☐ **Functions** ←
- ☐ Boolean expressions
- ☐ Conditionals
- ☐ Loops

Functions

False	await	**else**	**import**	pass
None	**break**	except	**in**	raise
True	class	finally	**is**	**return**
and	**continue**	**for**	lambda	try
as	**def**	**from**	nonlocal	**while**
assert	**del**	global	**not**	with
async	**elif**	**if**	**or**	yield

Today's topics

- ☑ Running Python – interactive and script mode
- ☑ Math
- ☑ Variables
- ☑ Types
- ☑ Text formatting
- ☑ Functions
- ☐ **Boolean expressions** ←
- ☐ Conditionals
- ☐ Loops

Boolean expressions

False	await	**else**	**import**	pass
None	**break**	except	**in**	raise
True	class	finally	**is**	**return**
and	**continue**	**for**	lambda	try
as	**def**	**from**	nonlocal	**while**
assert	**del**	global	**not**	with
async	**elif**	**if**	**or**	yield

Today's topics

- ☑ Running Python – interactive and script mode
- ☑ Math
- ☑ Variables
- ☑ Types
- ☑ Text formatting
- ☑ Functions
- ☑ Boolean expressions
- ☐ **Conditionals** ←
- ☐ Loops

Conditionals

False	await	else	**import**	pass
None	**break**	except	**in**	raise
True	class	finally	**is**	**return**
and	**continue**	**for**	lambda	try
as	**def**	**from**	nonlocal	**while**
assert	**del**	global	**not**	with
async	elif	if	**or**	yield

Today's topics

- ☑ Running Python – interactive and script mode
- ☑ Math
- ☑ Variables
- ☑ Types
- ☑ Text formatting
- ☑ Functions
- ☑ Boolean expressions
- ☑ Conditionals
- ☐ **Loops** ←

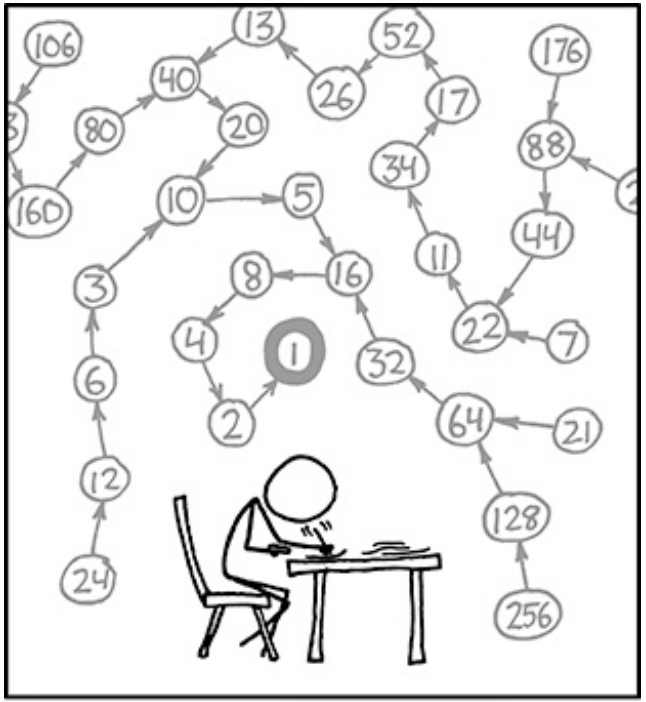
Loops

False	await	**else**	**import**	pass
None	**break**	except	**in**	raise
True	class	finally	**is**	**return**
and	**continue**	**for**	lambda	try
as	**def**	**from**	nonlocal	**while**
assert	**del**	global	**not**	with
async	**elif**	**if**	**or**	yield

problem_3_collatz.py

- Given a number:
  - If it's even, divide it by 2
  - If it's odd, multiply it by 3 and add 1
- Repeat until you reach 1
- Print the Collatz sequence for each number from 1 to 100.

Collatz conjecture



XKCD #710 by Randall Munroe

THE COLLATZ CONJECTURE STATES THAT IF YOU PICK A NUMBER, AND IF IT'S EVEN DIVIDE IT BY TWO AND IF IT'S ODD MULTIPLY IT BY THREE AND ADD ONE, AND YOU REPEAT THIS PROCEDURE LONG ENOUGH, EVENTUALLY YOUR FRIENDS WILL STOP CALLING TO SEE IF YOU WANT TO HANG OUT.

Alt text: The Strong Collatz Conjecture states that this holds for any set of obsessively-hand-applied rules.

Collatz conjecture on Wikipedia:
https://en.wikipedia.org/wiki/Collatz_conjecture

Today's topics

- ☑ Running Python – interactive and script mode
- ☑ Math
- ☑ Variables
- ☑ Types
- ☑ Text formatting
- ☑ Functions
- ☑ Boolean expressions
- ☑ Conditionals
- ☑ Loops

Covered Keywords – Week 1

False	await	**else**	**import**	pass
None	**break**	except	**in**	raise
True	class	finally	**is**	**return**
and	**continue**	**for**	lambda	try
as	**def**	**from**	nonlocal	**while**
assert	**del**	global	**not**	with
async	**elif**	**if**	**or**	yield

Next week's topics

- ❑ Dictionaries
- ❑ Lists
- ❑ Sets and tuples
- ❑ Handling exceptions
- ❑ Working with text files
- ❑ Working with CSV files
- ❑ Making HTTP requests
- ❑ Command line
- ❑ Script arguments

Covered Keywords – Week 2

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Course Project

Automated daily email

Good morning, Arianne 👋

Today's weather ☁️

Today there will be light snow.
High: 5.9 °C (42.6 °F)
Low: -0.9 °C (30.3 °F)

☀️ 7:48 AM
🌙 5:16 PM

Joke of the day 🤣

Little Johnny comes home from his first day of school. His mother asks, "So, what did you learn at school today?" Little Johnny replies, "NOT ENOUGH. They want me to come back tomorrow!"

Quote of the day 💬

When you recover or discover something that nourishes your soul and brings joy, care enough about yourself to make room for it in your life.
~ Jean Shinoda Bolen

Next game 🏀

In 2 days @ 7:30 PM ET
Home: Toronto Raptors
Away: Atlanta Hawks

Project

- Find some free APIs to retrieve data from
  - weather
  - quote/joke of the day
  - calendar, reminders, todos
  - sports, stocks, events

- Send email every morning

- Basic – plain text email
- Advanced – HTML (formatted) email

Homework

- Create email content text

Homework

- Find **3 APIs** you would like to try to incorporate
  - We'll try to get at least 1 to work
  - You can email me at arianne.dee.studios@gmail.com if unsure
- Create a **new Gmail account** to send your emails from
- Create an email with fake data for now
  - instructions on next slide

Homework

1. Copy your **greeting.py** file into **homework/homework_1.py**

2. Replace the multiple print statements with a ***multi-line f-string***; this will be your email message content

3. Move the temperature conversion to a function

4. Add content for the APIs that you want to use, with data:
   - Hardcoded
   - Input from the user
   - Random

Homework extras

- Feel free to add or edit it to make it personal to you:
  - Use Fahrenheit instead of Celsius as the default
  - Use a writing style you like and emoji
- Include content that you might want to read every morning
  - e.g. link to daily Wordle, crossword, web comic
- If you know HTML/CSS (or are ambitious) you can incorporate text styling, colour and images

Week 2 homework

- Accept the name input as a script argument
- Retrieve the reminder list from a text file
- Retrieve weather data from APIs
- Run script every morning

Week 3 homework

- Create a virtual environment for the project
- Send a text email (automated)
- Refactor into modules and functions (and classes)
- Optional: Send an HTML formatted email

Quiz

- Run **quizzes/quiz1.py** and answer it in the console

- There are links to relevant video lessons you can review if you get a question wrong

More practice

- Bonus practice problems you can try that incorporate the lessons from today's class
- Located in **practice/week_1/** folder
- You may need to do some simple internet searches to finish a portion of the problems

Video references for week 1

- **Running Python** – Intro to Python lesson 1.1
- **First code** – Intro to Python lesson 1.2
- **Variables** – Intro to Python lesson 2.2
- **Types** – Intro to Python lesson 2.1
- **Strings** – Intro to Python lesson 2.7
- **Functions** – Intro to Python lesson 2.6
- **Conditions** – Intro to Python lesson 3.2
- **Conditionals (if else)** – Intro to Python lesson 3.3
- **While loops** – Intro to Python lesson 4.1
- **For loops** – Intro to Python lesson 4.4

Supplemental videos

- **Week 1 review** – Next Level Python lesson 1.2
- **Dates and times** – Next Level Python lesson 1.5
- **Regular expressions** – Next Level Python lesson 1.6
- **Command line** – Next Level Python lesson 3.1
- **Git and GitHub**– Next Level Python lesson 3.4
- **Debugging** – Next Level Python lesson 6.1
- **Testing** – Next Level Python lesson 6.2
- **Intro to data analysis** – Intro to Python lesson 6
- **Intro to web apps** – Intro to Python lesson 7