coding temple

Python Basics

Module 2:

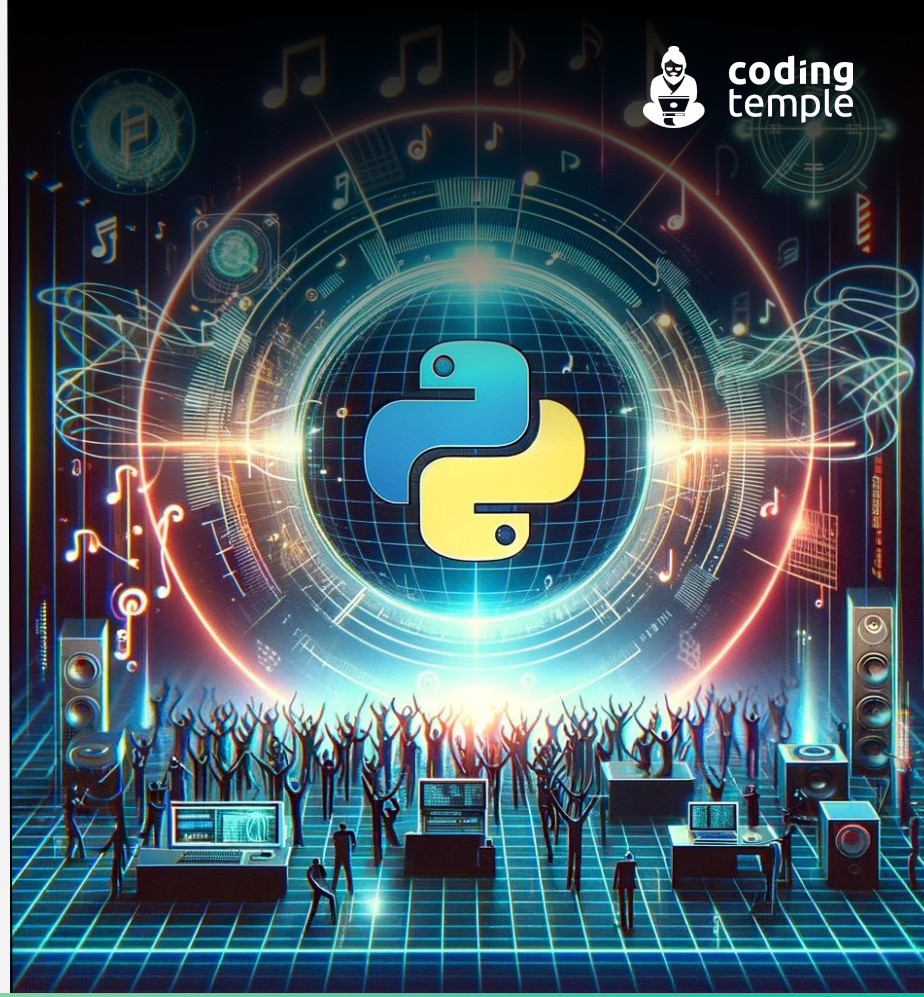
Chapter 1.2:

Lesson 1: Python Syntax

Chapter 1.2:

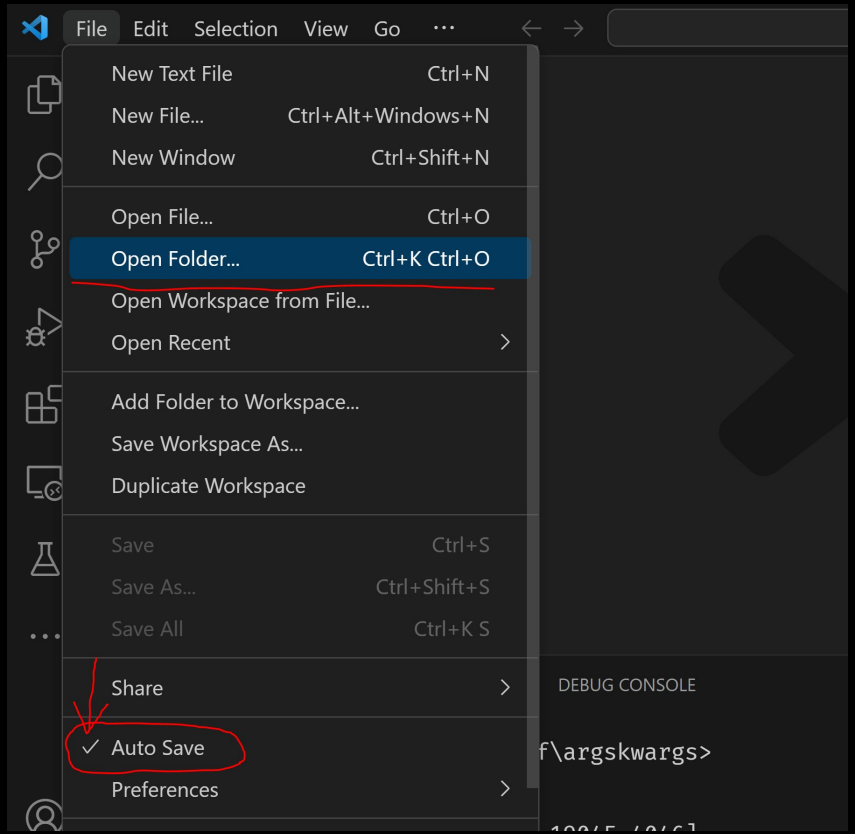
Grooving with Python: The Charm of Its Language and Syntax

Meeting Python, The Soulful Composer



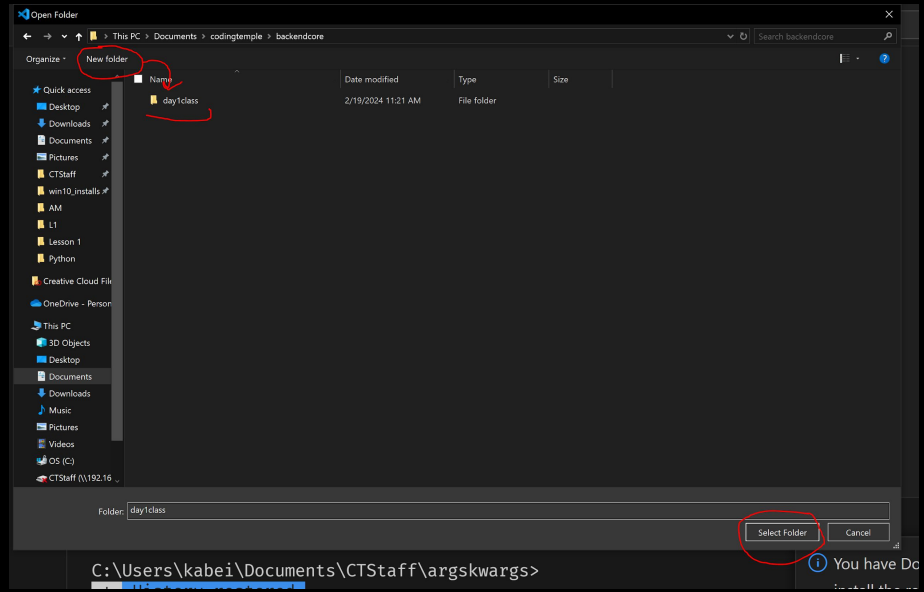
Getting Started with Python Code with VS Code

- Open VS Code and Turn on Auto Save
- Install Python Extension (if you haven't)
- File -> Open Folder
  Create a new folder for your project
  EVERY SINGLE PROJECT SHOULD HAVE ITS OWN FOLDER
- Open that newly created Folder
- Create a new file in this folder using the menu on the left and give it a name followed b the .py extension



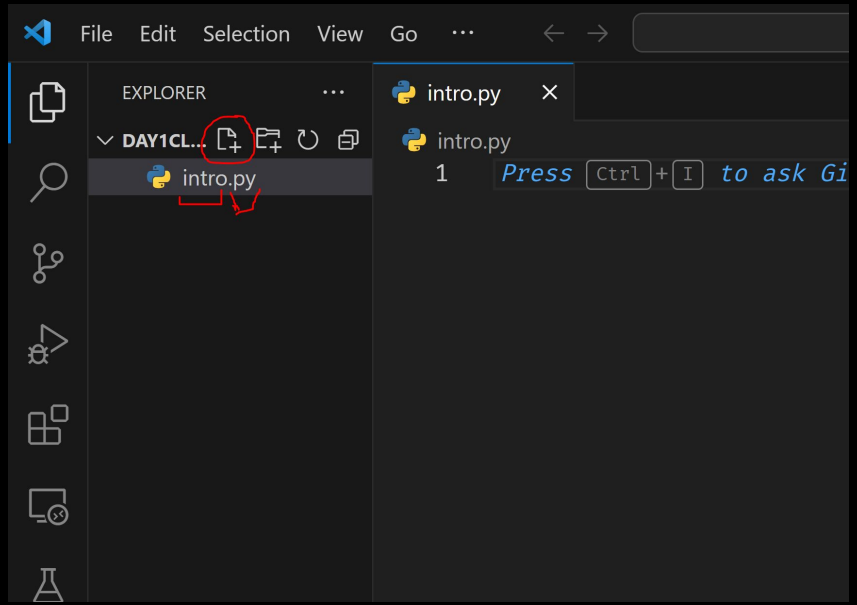
Getting Started with Python Code with VS Code

- Open VS Code and Turn on Auto Save
- Install Python Extension (if you haven't)
- File -> Open Folder
  - Create a new folder for your project
  - **EVERY SINGLE PROJECT SHOULD HAVE ITS OWN FOLDER**
- Open that newly created Folder
- Create a new file in this folder using the menu on the left and give it a name followed b the .py extension

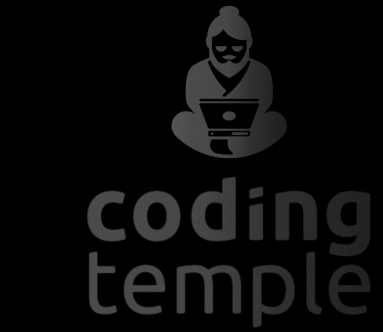


Getting Started with Python Code with VS Code

- Open VS Code and Turn on Auto Save
- Install Python Extension (if you haven't)
- File -> Open Folder
  Create a new folder for your project
  EVERY SINGLE PROJECT SHOULD HAVE ITS OWN FOLDER
- Open that newly created Folder
- Create a new file in this folder using the menu on the left and give it a name followed b the .py extension



Lets Practice



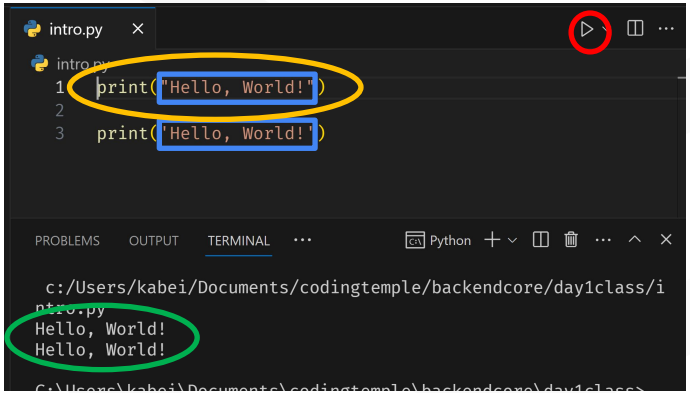
Printing text to the Terminal

Define text that you want to display to the user using a String datatype, You can do this by adding single or double quotes around your text

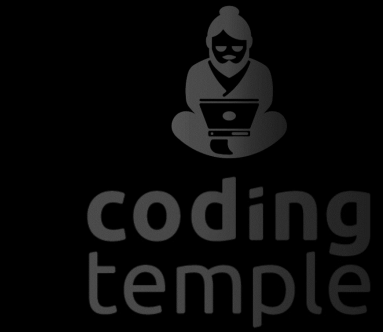
You can show this text in the terminal using the print function. The print function uses parathesis after the work print. You will put your String (text in quotes) inside the parathesis to tell the print function this is what you want to print

Press the play button to execute your Python code and have it run.
This can also be done in the terminal by typing 'python filename.py' and replacing the filename with the name of the file you are trying to run

After running your program you can see the output in the terminal



Lets Practice

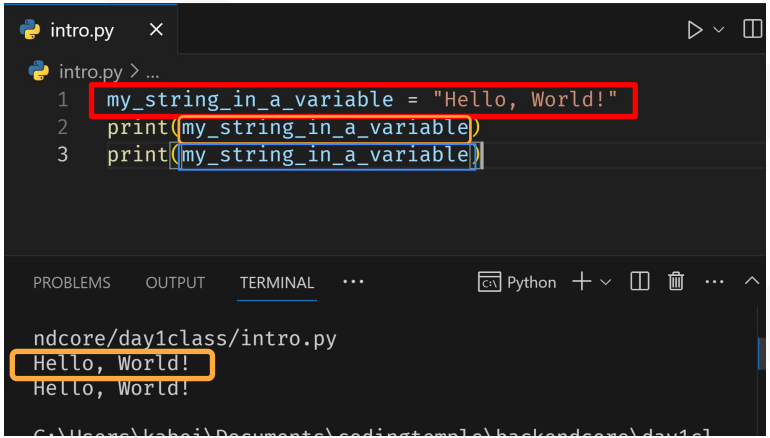


Saving String to Variables

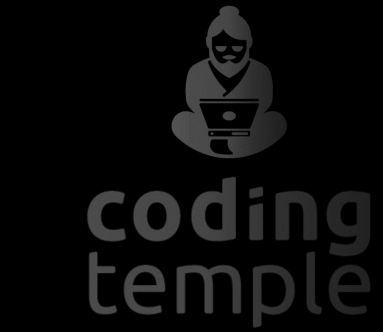
You can save data into variables using an equal sign. You can name the variable whatever you want, but there are a few rules (see next slide). The variable name always goes on the left side of the equal sign and then the data you are saving goes on the right

Instead of having to create your data over and over again, we can now call it by its variable name

We save things to variables to make it easy to reuse them and to call them again.



Lets Practice



Naming Convention: What's in a Name?

coding temple

RULES	DOs	DONTs
No variables starting with numbers	*variable1*	*1variable*
Snake_case: user_name, calculate_average.	_ Underscores	Spaces
Make all the sense possible	Relevant names	Unreal or names with no sense
No Symbols	jack_and_jill	jack&jill
No Python Reserved Word	my_break	break

Naming Convention: Reserved Words?

coding temple

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

Naming Convention: What's in a Name? (Cont.)

coding temple

Let´s refresh some concepts about The Importance of Good Naming Conventions:

1. **Snake Case:** Words are connected using underscores, and all letters are lowercase. For example, "coffee_beans."
   We use this for Python functions and variables.
   We use Uppercase Snake when creating constants (or variables whose data should never change)
   This is the main way you'll define variables in python

2. **Pascal Case:** All words start with a capital letter, and there are no spaces or underscores. For example, "CoffeeBeans."
   We use this in Python for Classes

3. **Camel Case:** The first word is lowercase, and subsequent words have their first letter capitalized. For example, "coffeeBeans." **We Do NOT use this convention in Python**.

intro.py > ...

```python
# Snake Case Variable name
user_name="Johnny"

# Snake Case use for function name
def hello_world():
    print("Hello World")

# Upper Case Snake Constant Variable name
CLASS_WEEKS=4

# Pascal Case use for class name
class MyFirstClass:
    pass

# Camel Case, not used in python,
# but used in many languages like
# C++ and JavaScript
myJavaScriptVariable="Hello World"
```

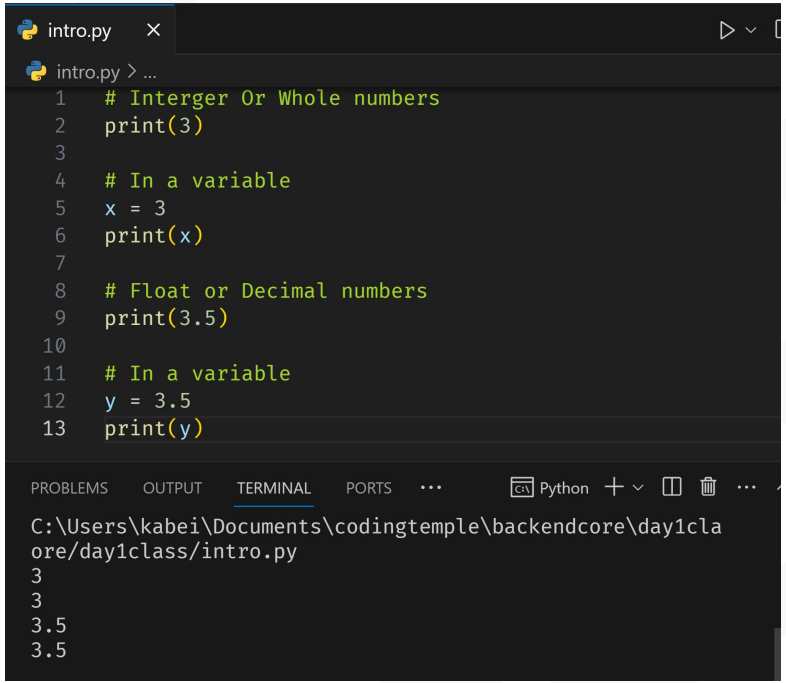
Numbers in Python

Numbers are divided into 2 groups whole numbers and decimal numbers. In python any number with a decimal will automatically be considered a decimal number

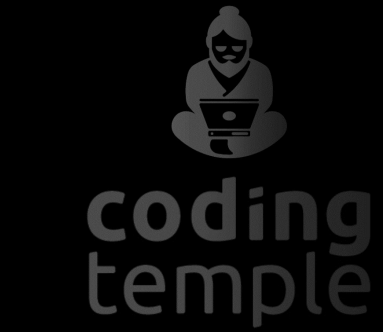
In programming we call these whole numbers Integers and the decimal numbers are called Floats (Floating Point Decimals)

Assigning Numeric Data to a variable is simple. You just type the number. Python will use Duck Typing to determine if the variable is a Float or an Int (Integer)

Duck Typing is the method python uses to determine the datatype of a variable (or what kind of data you are storing in a variable) The way to think about this is if it walks, talks and acts like a Duck it must be a duck



Lets Practice



The arithmetic operators help you handle all sorts of numerical data, from simple addition to finding remainders or squaring up with exponents.

NAME	SYMBOL	COMMON USES	EXAMPLE
Addition	+	• **Shopping Lists**: Combine items from multiple lists. • **Budgeting**: Sum up various expenses to get the total.	**Tossing strawberries and blueberries into a bowl for a fruity mix:**
Subtraction	-	• **Bank Transactions: Calculate balance after withdrawals.** • **Travel: Determine distance left to travel.**	**too much cheese and want just the right amount for your sandwich:**
Multiplication	*	• **Bulk Purchases: Calculate total cost for multiple items.** • **Gardening: Find the area covered by planting seeds at intervals.**	**too much cheese and want just the right amount for your sandwich:**
Division	/	• **Cooking: Divvy up ingredients for portion-controlled recipes.** • **Sports: Split total time among team members in a relay race.**	

```
strawberries = 10
blueberries = 15
mixed_fruits = strawberries + blueberries
print(mixed_fruits)  # Outputs: 25
```

```
initial_cheese_slice = 5
cheese_used = 2
cheese_left = initial_cheese_slice - cheese_used
print(cheese_left)  # Outputs: 3
```

```
pizza_slices_per_person = 2
party_guests = 5
total_slices_needed = pizza_slices_per_person * party_guests
print(total_slices_needed)  # Outputs: 10
```

```
pie_pieces = 8
people_at_table = 4
pieces_per_person = pie_pieces / people_at_table
print(pieces_per_person)  # Outputs: 2.0
```

Arithmetic Operators: The Math Chefs of Python's Kitchen (Cont.)

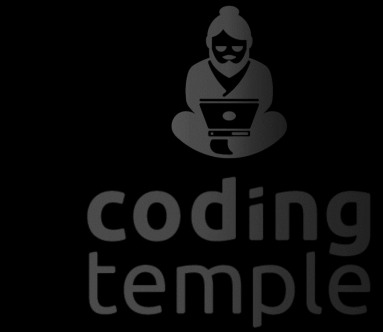
NAME	SYMBOL	COMMON USES	EXAMPLE
Modulus	%	• Baking: Ensure you never add more than the recipe's limit. • Divisible Numbers: To find out if a number is even or odd.	
Floor Division	//	• Crafting: Divide materials into whole units without leaving incomplete pieces. • Budgeting: Calculate how many items you can buy without exceeding a limit.	
Exponentiation	()	• Science: Model exponential growth or decay. • Finance: Calculate compound interest.	

```python
cookies = 14
cookies_per_jar = 5
outside_jar = cookies % cookies_per_jar
print(outside_jar)  # Outputs: 4
```

```python
total_sandwiches = 7
people = 3
sandwiches_each = total_sandwiches // people
print(sandwiches_each)  # Outputs: 2
```

```python
tea_strength = 2
cups_of_tea = tea_strength ** 2
print(cups_of_tea)  # Outputs: 4
```

Lets Practice

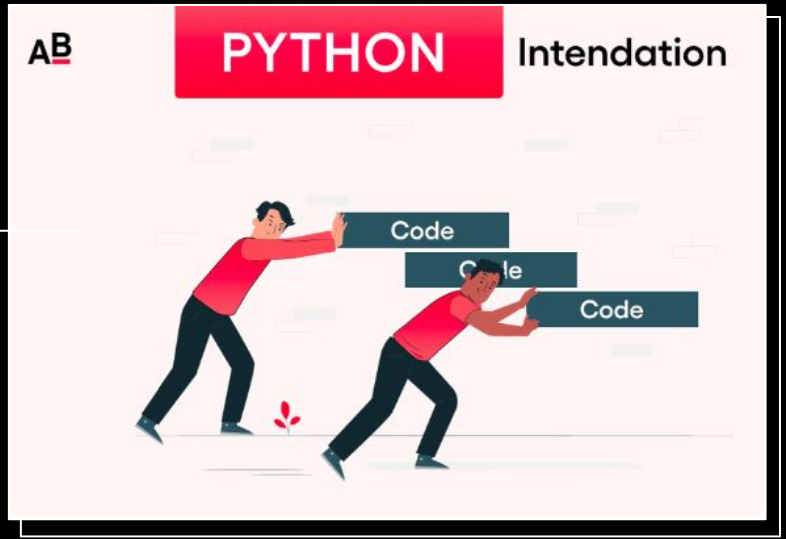


Refresher on Boolean (Comparison) Operators

Operator	Meaning
== (double equal to)	Equal to
<	Less than
>	Greater than
!=	Not equal to
<=	Less than or equal to
>=	Greater than or equal to

Python's Stylish Syntax: The Art of Indentation

1. **Indentation Defines Blocks:** Uses 4 spaces or 1 tab.
2. **Readability Focus:** Groups-related statements, enhancing clarity.
3. **Syntax Simplicity:** Clean, straightforward code structure.
4. **Attention to Detail:** Incorrect indentation leads to errors.



Let's analyzing the following exercise:

```
if 5 > 2:
    print("Five is greater than two!")
```

Remember, when we talked about The Comparison Operators in the Algorithm lesson in Intro Week?

This example is how we represent a conditional statement in Python!

⚠️Common Whitespace Errors ⚠️

coding temple

Indentation inconsistency example:

```
if True:
    print("This looks fine!")
  print("Oh no, a tab sneaked in!") # Imagine a tab before this line!
```

Unexpected Incident example:

```
print("All is well")
    print("Wait, why am I here?") # Oops! This line is wrongly indented.
```

Expected Incident example:

```
if True:
print("I should have been indented!") # This will raise an error.
```

Python Indentation Practice with if Statements

Code Correction

Correct the following code:

```
weather = "sunny"

if weather == "sunny":
print("Wear sunglasses!")
else:
print("Take an umbrella!")
```

Spotting Indentation Errors

Read the following code. Is the indentation correct?

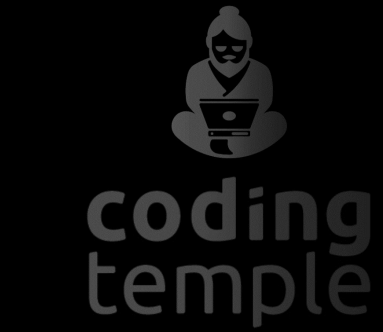
```
mood = "excited"

if mood == "excited":
    print("Yay! Let's have fun.")
    else:
print("Let's find something fun to do!")
```

Common Whitespace Errors (Cont.)

How to Dodge These Errors 🎯

- **Stay Consistent: spaces or tabs**
- **Use a Good Text Editor or IDE: they can convert tabs to spaces or vice versa**

Lets Practice



COMMENTS in Python

- **VALUABLE HINT! Use of Comments on your code:** # symbol to insert comments for better understanding and explanations.
  1. # **(Hash/Octothorp Symbol)**: Detonate Comments for the coder and anyone reading the code. They are ignored by the Python interpreter.
  2. print() **Function:** Show output contained in the String **(' ')** or double **(" ")** quotes.

Example of a comment inserted in a code:

```
# This is a comment
print("This is not a comment!")
```

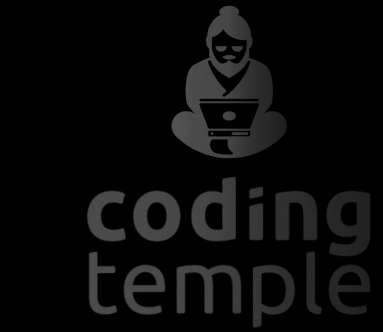
COMMENTS in Python

- **Multi-Line Statements: Write long and complex statements without making the code difficult to read.**

1. double quotes **(""")** and triple single quotes **(''')** create multi-line string literals. Used to create mutli-line strings for printing or that describe the purpose of a function or class.
2. **Backslash for Line Continuation**
3. **Spaces After Line Continuation**

```
a = '''Python's
multi-line
groove!'''

b = "Hello, Coding Templers! \
How are you today?"

print(a)
print(b)
```

Lets Practice

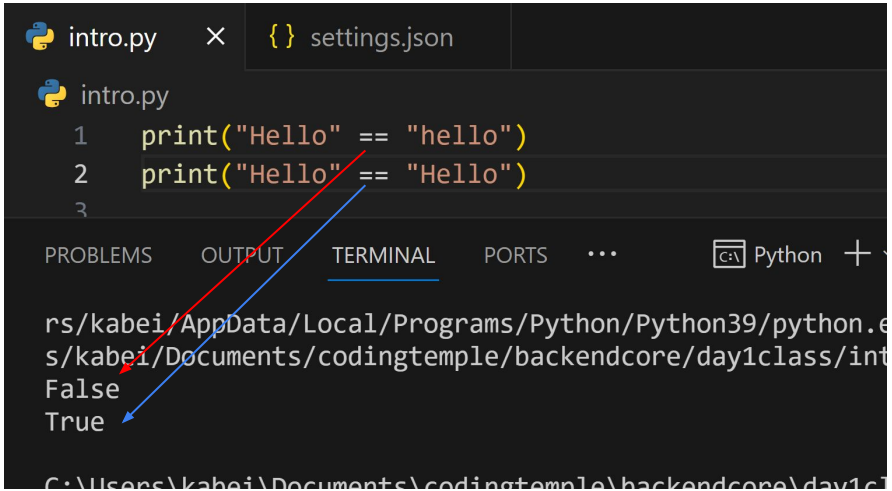


HeY YoU GuYs CaSiNG MatTErs!!!!

Casing Matters

When comparing if two words are the same Uppercase and Lowercase Letters matter.

Lower and Upper cased letter are treated as different letters, so HELLO and hello would not be said to be equal by Python



Case Sensitivity: The Shouting Game

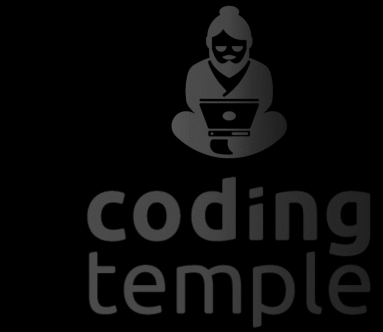
PYTHON is a "case-sensitive Language: It distinguishes between uppercase and lowercase letters.

Advantages	Drawbacks
Flexibility	Can lead to bugs
Encourages consistency when naming variables	Can be confusing

```
# Python Café where 'Coffee' and 'coffee' are two different things

Coffee = "Strong dark roast"
coffee = "Light cold brew"

print("Welcome to Python Café!")

print("\nIn Python:")
print("'Coffee' holds: " + Coffee)
print("'coffee holds: " + coffee)
```

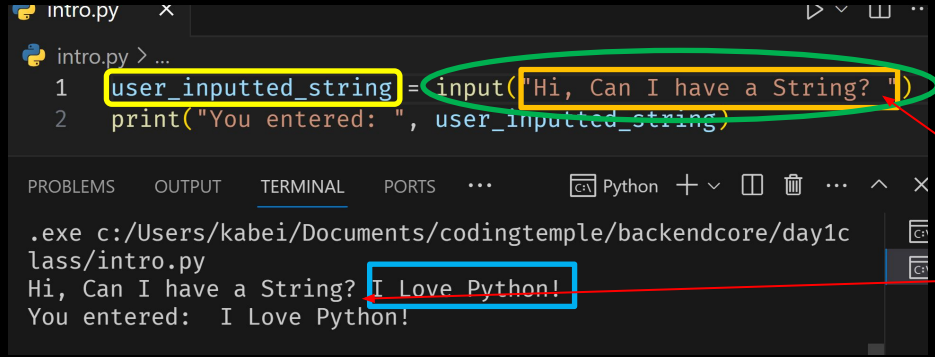
Strong dark roast
Light cold brew

Lets Practice



Asking for User Input via the Terminal

We can ask the User for a String to use in our program and assign it to a variable



The input function is like the print function but instead of printing it receives a string from the user and inside its parathesis you put the prompt for the user

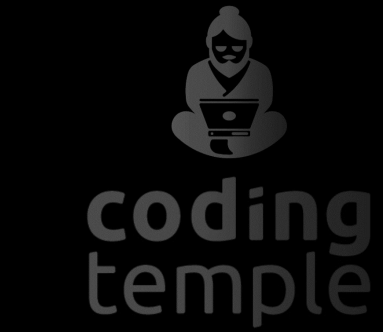
The prompt is printed to the user

We can the user response by assigning the returned value from the input function to a variable

The user replies to the Prompt in the Terminal “I Love Python!” was typed in by the user of our program

Note: The space at the end of my prompt string, This is here to give space between my question and the user input

Lets Practice



In Class Exercise: Your Mood Today

Ask the user how they feel today.

- **If they say "happy", print "That's great to hear!"**
- **if they say "sad", print "I hope your day gets better!"**

Ensure your if statement is properly indented.

Type Casting

```
1  my_int_as_string = "12"
2  print(my_int_as_string)
3  print(type(my_int_as_string))
4
5  my_int = int(my_int_as_string)
6  print(my_int)
7  print(type(my_int))
```

PROBLEMS 2 OUTPUT TERMINAL PORTS

```
s/kabei/Documents/codingtemple/backend
12
<class 'str'>
12
<class 'int'>
```

There is a variable that contains the number 12 but since it is in quotes, python will treat this as text. If we want to be able to treat this as an integer number, (without redeclaring the variable as an int) we will need to cast it as such.

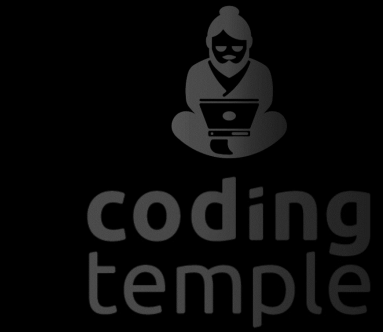
The type function returns back the datatype that python know the variable to be.

The int() function will convert the data passed to it to an integer value, if the value makes sense to turn to an integer.

We can see the output of the type function which shows the conversion from string type to integer has taken place.

Converting datatypes is known as casting. There are many functions that allow for casting today we will talk about str, int, and float

Lets Practice



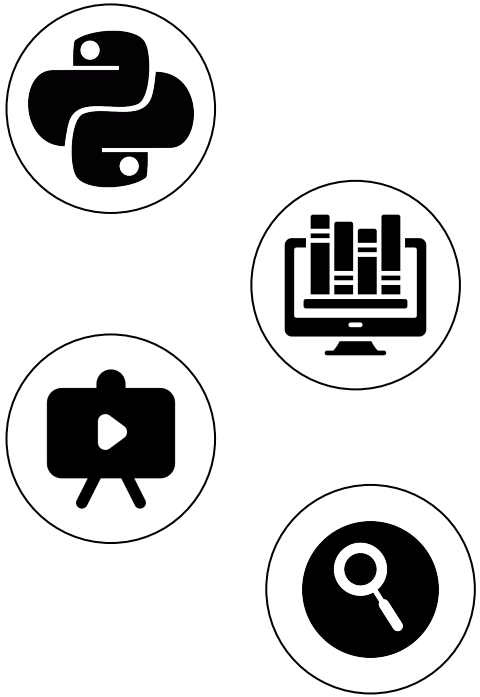
Why is the Python Documentation So Crucial?

- Direct from the Source
- Comprehensive & Up-to-Date
- Problem Solver
- Enhances Understanding

Step - by Step to gain the most of Python Documentation

- **The Landing Page:** Python's official documentation page: Your place for Python
- **Tutorial:** Guidance no matter what!
- **Library Reference:** Where everything is there for you.
- **The Search Box:** Python at your fingertips!

HINT: While coding, if you need a quick reference on a function or module, just type "Python docs" in your favorite search engine. For instance, "Python docs list comprehensions"



Other Python Documentation

W3Schools Python Tutorial - Your Beginner's Paradise: W3Schools Python Tutorial. Try It Yourself' editor.

Real Python - Deep Dives into Python's Depths: Real Python: From Basics to Advanced explorations.

Geeks for Geeks Python Programming Language - Unleash Your Python Prowess: Geeks for Geeks Python Programming Language. All about algorithms with data structures and more.

Corey Schafer's Python Tutorials - Python Through the Lens, with Class!: Corey Schafer's YouTube Channel: Interactive place to play, Simplified complex topics and provides an enjoyable learning experience.

Exercise

Visit
https://docs.python.org/3.10/tutorial/introduction.html

Spend some time reading through this tutorial then share with the class one thing you learned from the documentation.

Errors are our Friends

ERRORS... The best thing about committing them is what they can offer us!

coding temple

SYNTAX ERRORS	LOGIC ERRORS	EXCEPTIONS
•Missing Parentheses in Call to 'print' •Mismatched or Missing Parenthesis, Brace, or Bracket •Missing Colon •Improper Indentation •Mismatched Quotes •Using a Reserved Keyword •Missing = in Assignment •Unexpected Indent •EOL Error While Scanning String Literal •Forgetting to Import a Library or Function	•Incorrect Loop Logic •Misplaced Indentation •Off-by-One Error •Incorrect Conditionals •Failing to Initialize Variables •Misunderstanding Scope •Wrong Algorithm or Formula •Not Accounting for Edge Cases	•ZeroDivisionError •TypeError •ValueError •NameError •IndexError •KeyError •FileNotFoundError •AttributeError •ImportError •OverflowError

Click Next

Common Errors: Stay Calm Code On (1/4)

ZeroDivisionError

Explanation: This error occurs when a number is divided by zero. Division by zero is undefined in mathematics, and Python raises this error to signal the illegal operation.

Type of Error: This is usually a logic error, as the code's logic should prevent division by zero from occurring.

TypeError

Explanation: A TypeError is raised when an operation is applied to an object of an inappropriate type. This can happen if you try to perform an operation on a type that doesn't support it, like adding a string and an integer.

Type of Error: This can be a typo error if you accidentally used the wrong variable or a logic error if the operation's appropriateness wasn't properly considered.

ValueError

Explanation: This error occurs when a function receives an argument of the correct type but with an inappropriate value. For example, converting a string that doesn't represent a number to an integer will raise this error.

Type of Error: It is often a logic error because the value passed to a function doesn't make sense, even though it's of the correct type.

Common Errors: Stay Calm Code On (2/4)

NameError

Explanation: This error is raised when a local or global name is not found. This can occur if a variable is used before it is defined or if there is a typo in the variable name.

Type of Error: This can be a typo error (misspelling a variable name) or a logic error (using a variable before it is defined).

IndexError

Explanation: An IndexError is raised when attempting to access an index that is out of the bounds of a sequence type, like a list or a tuple.

Type of Error: It's generally a logic error, as the code should ensure that index access is within the valid range of the sequence.

KeyError

Explanation: This error occurs when a dictionary key is accessed that does not exist in the dictionary.

Type of Error: It can be a typo error if the wrong key is accidentally used, or a logic error if the code incorrectly assumes the presence of a key.

Common Errors: Stay Calm Code On (3/4)

FileNotFoundError

Explanation: Raised when an attempt to open a file (or a similar file-like object) fails because the file does not exist.

Type of Error: This could be a typo error if the file name or path is misspelled, or a logic error if the code assumes the existence of a file without verifying it.

AttributeError

Explanation: An AttributeError is raised when an attribute reference or assignment fails, such as when trying to access a method or property of an object that doesn't exist.

Type of Error: This can be a typo error if the attribute name is misspelled, or a logic error if the code wrongly assumes the object has a particular attribute or method.

ImportError

Explanation: This error occurs when the **import** statement cannot find the module definition. It can also occur when **from ... import ...** fails to find a name that is supposed to be defined in a module.

Type of Error: Often a typo error if the module name is incorrect, or it could be an environment error if the module is not installed in the current environment.

Common Errors: Stay Calm Code On (4/4)

OverflowError

- **Explanation**: Raised when an arithmetic operation exceeds the limits of a numeric type, resulting in an "overflow". This is more common with fixed-size numeric types (like C longs) than with Python's long integers.
- **Type of Error**: It's generally a logic error, particularly in algorithms dealing with large numbers, where the possibility of exceeding numerical limits wasn't properly accounted for.

Lets Practice