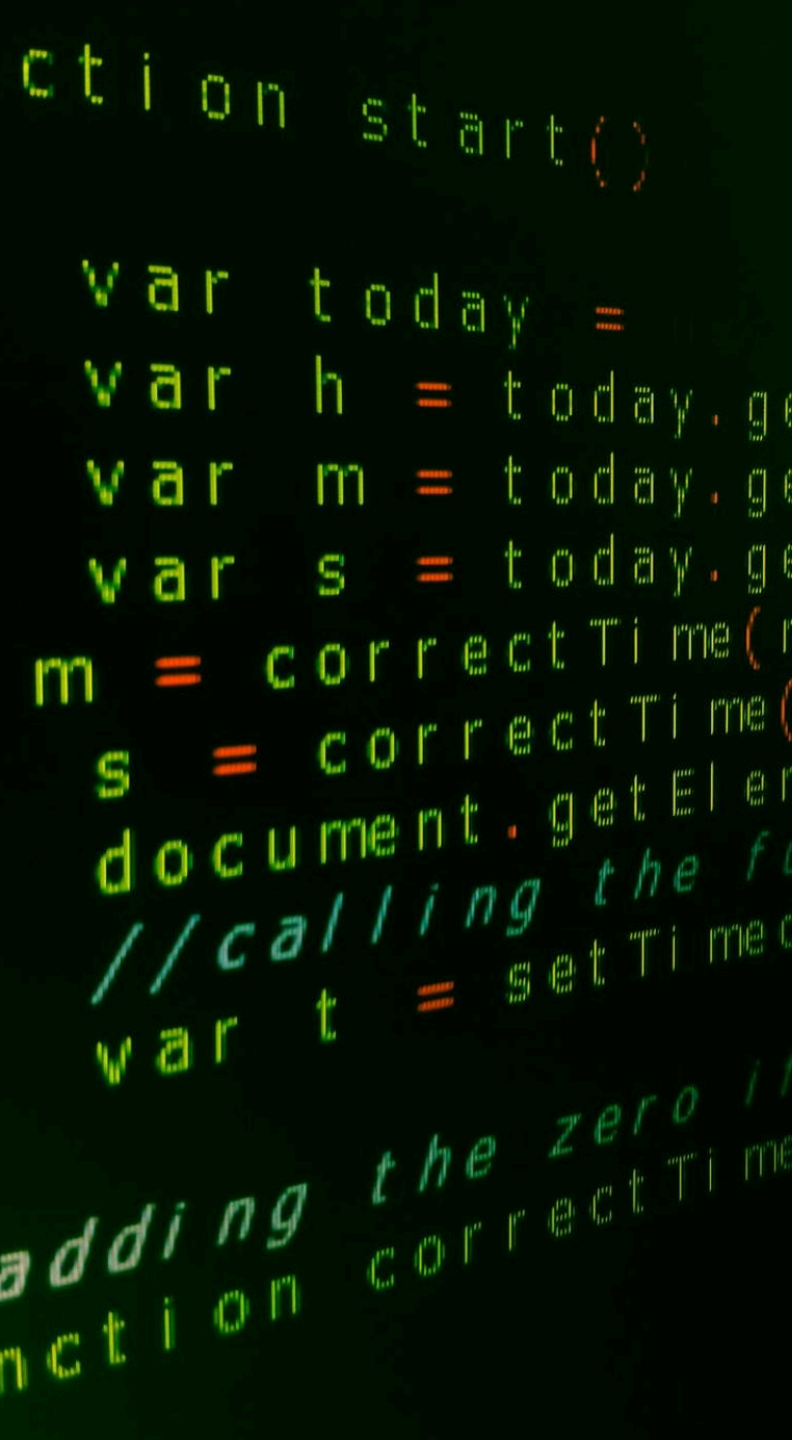
JANUARY 17TH, 2025

Variables, Expressions, and Conditionals

CE 311K - L03



Review: Data Types

Data Types specify the kind of data stored

Python uses **Dynamic Typing** - don't have to specify data type

```
i = 1              # int
pi = 3.141         # float
fname = "Hagen"    # str (double quotes)
lname = 'Fritz'    # str (single quotes)
active = True      # bool
```

Variable Naming - General

Use **Descriptive** and **Meaningful** names

```
total_cost = 100.00 # versus `tc`
user_age = 31 # versus `ua`
```

Find a naming convention/style you like and **stick** with it

Snake Case, Camel Case, Pascal Case, and All Caps

Cannot start with numbers or use special characters

Except for underscores

Avoid using **Reserved** words

Variable Naming - Python

Python follow the **PEP 8** style guide

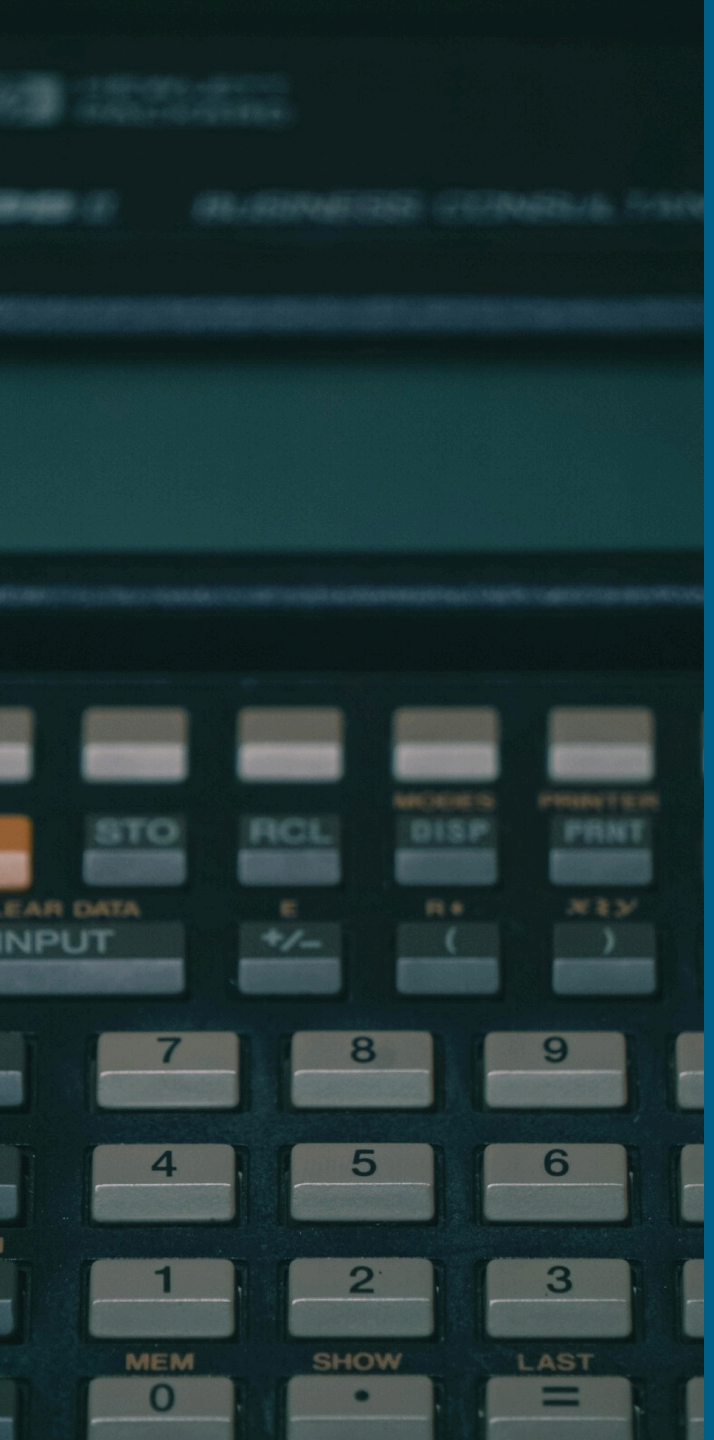
Snake case for variable and function names: *my_var, calculate_total()*

Pascal case for class names: *MyClass, StudentRecord*

All Caps for constants: *PI, MAX_USERS*

Variable names are **case-sensitive**

```
name = "Hagen"
Name = "Fritz"
print(name)      # will output Hagen
print(Name)      # will output Fritz
```



Arithmetic Expressions

Combine variables and numbers with **operators**

```
x = 3
y = x + 1 # y is 4
```

Python expressions are evaluated following **PEMDAS**

```
result = 4 * (2 + 1) ** 2 # 36
```

Implicit casting when dividing or combining *ints* and *floats*

```
num = 7 / 2 # 3.5
total = 10 + 2.5 # 12.5
```

Special Operators

You can perform **integer division** with //

```
value_float = 7 / 2  # 3.5
value_int = 7 // 2   # 3
```

Get the remainder using the **modulus** operator

```
remaining = 10 % 3 # 1
```

Update values concisely using **compound assignment operators**

```
num = 2
num += 3  # num is now 5
num *= 2  # num is now 10
```

Boolean Expressions

Comparison Operators: >, <, ==, >=, <=, *!=*

```
x = 2
y = 1
is_equal = x == y   # False
is_greater = x > y  # True
```

Logical Operators: *and, or, not*

```
conditions = (x != y) and (x > y) # True
```

Conditionals

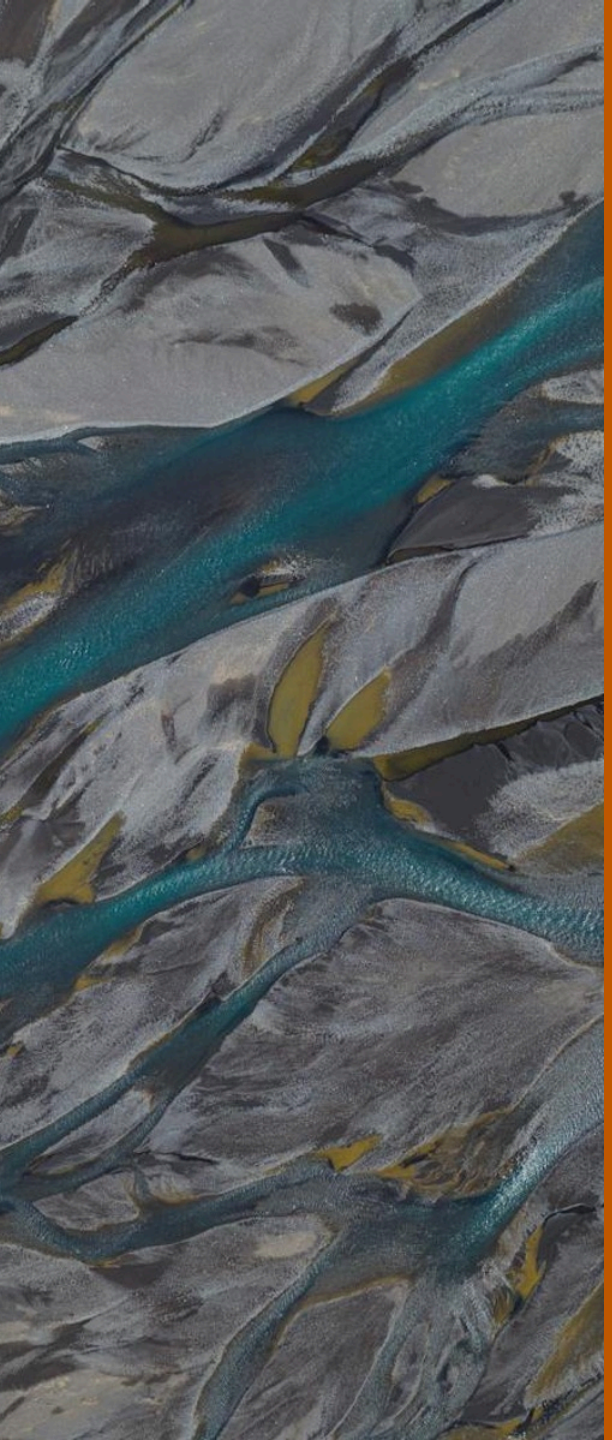
Conditionals allow us to execute code blocks based on conditions

```
temperature = 75.0
if temperature > 80.0:
    print("It's warm!")
```

Indentation is how Python knows what code to execute if the condition is true/false

```
age = 20
if age >= 18:
    print("You are an adult.") # only runs if condition is true

print("Thank you for using the age checker!") # Runs regardless
```



Multiple Conditions

We can provide a **catch-all** in case the condition is false: *else*

```
if is_rainy:
    print("It is raining")
else:
    print("No rain right now")
```

We can provide more conditions to check with *elif*

```
if temperature > 75:
    print("It is warm!")
elif temperature > 65:
    print("It is pleasant!")
else:
    print("It is cold")
```

Summary

Use **Descriptive** and **Meaningful** variable names

Use consistent styling

Avoid built-in words

Don't start with numbers/special characters

Arithmetic expressions assign values using **operators**

Expressions follow PEMDAS

Modulus (%) for remainder and **Floor Division** (//) for integer division

Compound Assignment Operators provide shorthand for updating

Boolean expressions use **Conditional** and **Logical Operators**

Conditionals dictate the flow of code

Indentation shows which code is executed

Catch-all *else* block

Define multiple conditions with *elif*