DATA TYPES AND EXPRESSIONS

Outline

- Operations
  - String Manipulation
- Expressions
  - Assignment
  - Mathematical
  - Boolean

Variables and Data Types

- Variables
  - Stores information your program needs
  - Each has a unique name
  - Each has a specific type that Python infers

Python simple type	what it stores	example values	operations
`int`	integer values	`42` `1234`	add, subtract, multiply, divide, remainder, compare
`float`	floating-point values	`9.95` `3.0e8`	add, subtract, multiply, divide, remainder, compare
`str`	sequence of characters	`"Hello world!"` `'I love this!'`	concatenate, and more
`bool`	truth values	`True` `False`	and, or, not

Data Types: Strings

Python type	what it stores	example values	operations
`str`	sequence of characters	`"Hello world!"` `'I love this!'`	+, len(), center(), count(), endswith(), find(), isalpha(), isacii(), isdigit(), islower(), isspace(), isupper(), ljust(), lower(), replace(), rjust(), split(), startswith(), strip(), swapcase(), title(), upper()

- Just a sampling of operations is listed – there are more. You need not memorize this list, but it should be useful when you want to work with strings in your programs.

- Demo code: StringDemo.py

Data Types: Integers

Python type	what it stores	example values	operations
`int`	integer values	42 1234	add, subtract, multiply, divide, remainder, compare

```
i = 23
j = 5

print("Addition:", (i + j))
print("Subtraction:", (i - j))
print("Multiplication:", (i * j))
print("Division:", (i / j))
print("Remainder:", (i % j))
print("i greater than j?", (i > j))
print("i less than j?", (i < j))
print("i equal to j?", (i == j))
print("i not equal to j?", (i != j))
```

Data Types: Floating Point Numbers

Python type	what it stores	example values	operations
`float`	floating-point values	`9.95` `3.0e8`	add, subtract, multiply, divide, compare

```
i = 23.5
j = 5.5

print("Addition:", (i + j))
print("Subtraction:", (i - j))
print("Multiplication:", (i * j))
print("Division:", (i / j))
print("i greater than j?", (i > j))
print("i less than j?", (i < j))
print("i equal to j?", (i == j))
print("i not equal to j?", (i != j))
```

Booleans

`not a` → "Is a set to `false`?"
`a and b` → "Are both a *and* b set to `true`?"
`a or b` → "Is either a *or* b (or both) set to `true`?"

a	b	a and b	a or b
false	false	**false**	**false**
false	true	**false**	**true**
true	false	**false**	**true**
true	true	**true**	**true**

a	not a
true	**false**
false	**true**

Data Types: Boolean / Logical

Python type	what it stores	example values	operations
`boolean`	truth values	`True` `False`	and, or, not

```
t = True
f = False

print("And:", t, "and", t, "is", (t and t))
print("And:", t, "and", f, "is", (t and f))
print("And:", f, "and", t, "is", (f and t))
print("And:", f, "and", f, "is", (f and f))
print()

print("Or:", t, "or", t, "is", (t or t))
print("Or:", t, "or", f, "is", (t or f))
print("Or:", f, "or", t, "is", (f or t))
print("Or:", f, "or", f, "is", (f or f))
print()

print("Not", t, "is", (not t))
print("Not", f, "is", (not f))
```

Boolean Expressions: Comparisons

- Given two numbers → return a `boolean`

operator	meaning	true example	false example
==	equal	`7 == 7`	`7 == 8`
!=	not equal	`7 != 8`	`7 != 7`
<	less than	`7 < 8`	`8 < 7`
<=	less than or equal	`7 <= 7`	`8 <= 7`
>	greater than	`8 > 7`	`7 > 8`
>=	greater than or equal	`8 >= 2`	`8 >= 10`

Is the sum of a, b and c equal to 0?
```
(a + b + c) == 0
```

Is grade in the B range?
```
(grade >= 80.0) and (grade < 90.0)
```

Is sumItems an even number?
```
(sumItems % 2) == 0
```

Leap Year Example

- Years divisible by 4 but not by 100 → leap year
- Years divisible by 400 → leap year

```
year = int(input("Enter the year: "))
isLeapYear = False

# Leap year if divisible by 4 but not by 100
isLeapYear = (year % 4 == 0) and (year % 100 != 0)

# But also leap year if divisible by 400
isLeapYear = isLeapYear or (year % 400 == 0)
print(isLeapYear)
```

Mathematical Expressions: Parentheses and Precedence

- Parentheses can change the order in which arithmetic operations are performed
  - examples:

```
(cost + tax) * discount
(cost + (tax * discount))
```

- Without parentheses, an expressions is evaluated according to the ***rules of precedence***, with the lowest precedence listed at the top.

Operator	Description
or	Boolean OR
and	Boolean AND
not x	Boolean NOT
<, <=, >, >=, !=, ==	Comparisons, including membership tests and identity tests
+, -	Addition and subtraction
*, /, %	Multiplication, division, remainder
**	Exponentiation

Type Conversion Quiz

expression	resulting type	resulting value
`int(3.14159)`		
`round``(3.6)`		
`2 * 3.0`		
`2 * int(3.0)`		
`int(2) * 3.0`		

Type Conversion Quiz

expression	resulting type	resulting value
`int(3.14159)`	`int`	`3`
`round``(3.6)`	`float`	`4`
`2 * 3.0`	`float`	`6.0`
`2 * int(3.0)`	`int`	`6`
`int(2) * 3.0`	`float`	`6.0`

String Conversion Quiz

expression	resulting type	resulting value
`int("30")`		
`float("30")`		
`int("30.1")`		
`float("30.1")`		
`int("$30")`		
`float(3.14)`		

String Conversion Quiz

expression	resulting type	resulting value
`int("30")`	`int`	`30`
`float("30")`	`float`	`30.0`
`int("30.1")`	`ValueError: invalid literal for int()`	
`float("30.1")`	`float`	`30.1`
`int("$30")`	`ValueError: invalid literal for int()`	
`float(3.14)`	`float`	`3.14`

Concatenating Strings with Other Types

```
s = ""
r = s

s = s + str(3 + 4)
r = s + str(2)

print(s)
print(r)

print(r + " Trombones")
print(r + str(3.41512))
print(str(3.41512) + r)
```

Double quotes with nothing in between, an empty String

String Concatenation Quiz

expression	resulting type	resulting value
`"testing " + str(1) + str(2) + str(3)`		
`"3.1" + 4159`		
`"2" + " + " + "3"`		
`str(1 + 2 + 3) + "66"`		

String Concatenation Quiz

expression	resulting type	resulting value
`"testing " + str(1) + str(2) + str(3)`	`String`	`"testing 123"`
`"3.1" + 4159`	`TypeError`	
`"2" + " + " + "3"`	`String`	`"2 + 3"`
`str(1 + 2 + 3) + "66"`	`String`	`"666"`

Summary

- Operations
  - String Manipulation
- Expressions
  - Assignment
  - Mathematical
  - Boolean