Introduction to Problem Solving in Python

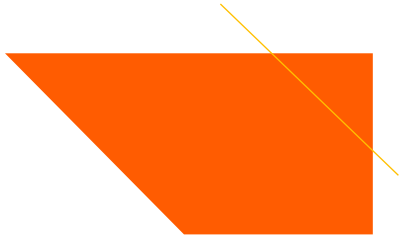
COSI 10A

Review: Order of operations

- When one operator appears in an expression, the order of evaluation depends on the **rules of precedence**

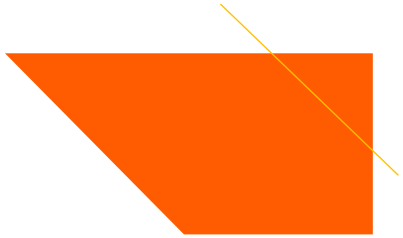
		Example	
( )	Parentheses	2 * (3 – 1) is 4	highest
**	Exponentiation	3 * 2**3 is 24	
*, /, //, %	Multiplication, Division, Modulus	7 // 2 * 4 is 12 7 // 3 % 3 is 2	
+, –	Addition, Subtraction	2 + 7 // 3 is 4	lowest

- Operators with the same precedence are evaluated from left to right (except exponentiation)



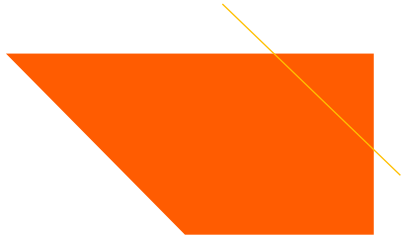
Review: Precedence

- What values result from the following expressions?
  - `9 // 5`
  - `695 % 20`
  - `7 + 6 * 5`
  - `7 * 6 + 5`
  - `248 % 100 / 5`
  - `6 * 3 - 9 // 4`
  - `(5 - 7) * 2 ** 2`
  - `6 + (18 % (17 - 12))`



Class objectives

- Variables
- Python keywords
- Errors

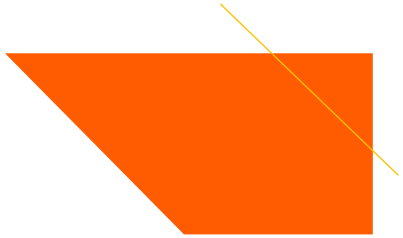


Variables

Example: receipt question

- What's bad about the following code?

```
# Calculate total owed, assuming 8% tax / 15% tip
print("Subtotal:")
print(38 + 40 + 30)

print("Tax:")
print((38 + 40 + 30) * .08)

print("Tip:")
print((38 + 40 + 30) * .15)

print("Total:")
print(38 + 40 + 30 + (38 + 40 + 30) * .15 + (38 + 40 + 30) * .08)
```



Example: receipt question

- What's bad about the following code?

```
# Calculate total owed, assuming 8% tax / 15% tip
print("Subtotal:")
print(38 + 40 + 30)

print("Tax:")
print((38 + 40 + 30) * .08)

print("Tip:")
print((38 + 40 + 30) * .15)

print("Total:")
print(38 + 40 + 30 + (38 + 40 + 30) * .15 + (38 + 40 + 30) * .08)
```

- The subtotal expression (38 + 40 + 30) is repeated
- So many print statements

Variables

- A **variable** is a piece of the computer's memory that is given a name and type, and can store a value
- Steps for using a variable:
  - **Declare/initialize it** - state its name and type and store a value into it
  - **Use it** - print it or use it as part of an expression

Declaration and Assignment

- **Variable declaration and assignment:** Sets aside memory for storing a value and stores a value into a variable
  - Variables **must be declared** before they can be used
  - The value can be an expression. The variable will store its result

- Syntax:

```
name = expression
```

- Example:

```
zipcode = 90210
myGPA = 1.0 + 2.25
```

zipcode	90210

myGPA	3.25

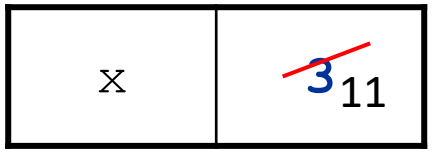
Using variables

- Once given a value, a variable can be used in expressions:

```
x = 3                  # x is 3
y = 5 * x - 1          # now y is 14
```

- You can assign a value more than once:

```
x = 3                  # 3 here

x = 4 + 7              # now x is 11
```

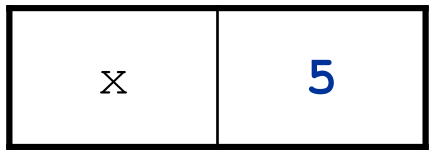


Assignment and algebra

- Assignment uses **=** , but it is not an algebraic equation.
  - **=** means, *"store the value at right in variable at left"*
  - The right side expression is evaluated first, and then its result is stored in the variable at left

- What happens here?

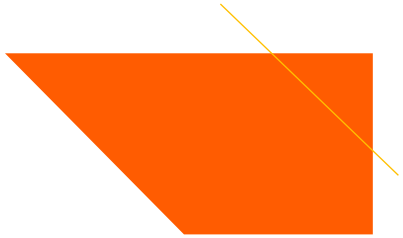
```
x = 3
x = x + 2    # ???
```



Receipt question ver. 1

```
def main():
    # Calculate total owed, assuming 8% tax / 15% tip
    print("Subtotal:")
    print(38 + 40 + 30)

    print("Tax:")
    print((38 + 40 + 30) * .08)

    print("Tip:")
    print((38 + 40 + 30) * .15)

    print("Total:")
    print(38 + 40 + 30 + (38 + 40 + 30) * .15 + (38 + 40 + 30) * .08)

main()
```

- Improve this program using variables



Receipt question ver. 2

```
def main():
    # Calculate total owed, assuming 8% tax / 15% tip
    subtotal = 38 + 40 + 30 #int

    print("Subtotal:")
    print(subtotal)

    print("Tax:")
    print(subtotal * .08)

    print("Tip:")
    print(subtotal * .15)

    print("Total:")
    print(subtotal + (subtotal) * .15 + (subtotal) * .08)

main()
```

- Happy?

Receipt question ver. 3

```
def main():
    # Calculate total owed, assuming 8% tax / 15% tip
    subtotal = 38 + 40 + 30 #int
    tax = subtotal * .08 #float
    tip = subtotal * .15 #float

    print("Subtotal:")
    print(subtotal)

    print("Tax:")
    print(tax)

    print("Tip:")
    print(tip)

    print("Total:")
    print(subtotal + tip + tax)

main()
```

- What about now?

Printing a variable's value

- Use a comma to print a string and a variable's value on one line

```
grade = (95.1 + 71.9 + 82.6) / 3.0
print("Your grade was", grade)
```

```
students = 11 + 17 + 4 + 19 + 14
print("There are", students,
      "students in the course.")
```

Output:

```
Your grade was 83.2
There are 65 students in the course.
```

Receipt question ver. 4

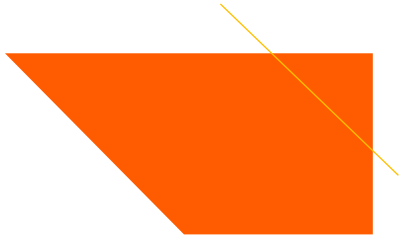
```
def main():
    # Calculate total owed, assuming 8% tax / 15% tip
    subtotal = 38 + 40 + 30 #int
    tax = subtotal * .08 #float
    tip = subtotal * .15 #float
    total = subtotal + tax + tip   # float



    print("Subtotal:", subtotal)
    print("Tax:", tax)
    print("Tip:", tip)
    print("Total:", total)

main()
```

```
    print(subtotal + tip + tax)

main()
```

Variables names

- Choose names for the variables that are meaningful
- They can be arbitrarily long
- They can contain both letters and numbers, **but they have to begin with a letter**
  - Convention: start variable name with lowercase letter
- Names with multiple words are separated with by the underscore character _
  - `my_name area_of_circle`
- If you give a variable an illegal name you get a syntax error

Keywords



Keywords

- A **keyword** is an identifier that you cannot use because it already has a reserved meaning in Python

```
and       del       from      not       while
as        elif      global    or        with
assert    else      if        pass      yield
break     except    import    print
class     exec      in        raise
continue  finally   is        return
def       for       lambda    try
```

String operations

- In general, you can't perform mathematical operations on strings (even if the string looks like a number)
  - Illegal: '2' - '1'
- The + operator works with strings. It performs **string concatenation**

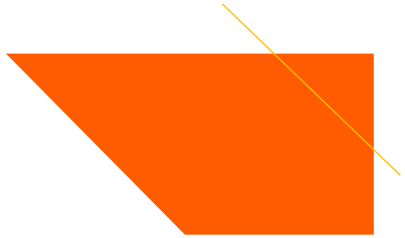
```
>>> first = "hello"
>>> second = "ciao"
>>> print(first + second)
```

Output: `hellociao`

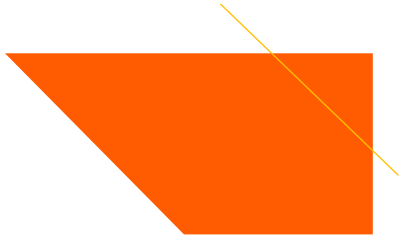
- The * operator also works. It performs **repetition**

```
>>> "ciao " * 3
```

Output: `ciao ciao ciao`



Debugging



Errors

- **Syntax errors** you are likely to make at this point:
  - Illegal variable name
  - Space between variables names e.g. `bad name = 5`

- **Runtime errors** you are likely to make at this point: “Use before defined”

```
>>> principal = 327.68
>>> interest = principle * rate
```