CSC 171 - Introduction to Computer Programming

Lecture #1 - Getting Started: An Introduction to Programming in Python

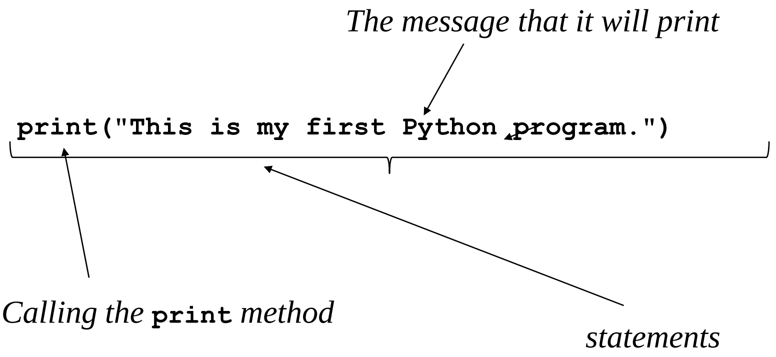
What Is Programming?

- Computers cannot do all the wonderful things that we expect without instructions telling them what to do.
- ***Program*** – a set detailed of instructions telling a computer what to do
- ***Programming*** – designing and writing computer programs
- ***Programming language*** – a language used to express computer programs.
  - We will be learning the Python programming language

The Python Programmng Language

- Python is an interpreted language
- interpreted means that Python looks at each instruction, one at a time, and turns that instruction into something that can be run.
- That means that you can simply open the Python interpreter and enter instructions one-at-a-time.
- You can also import a program which causes the instructions in the program to be executed, as if you had typed them in.
- To rerun an imported program you reload it.

A First Program



The message that it will print

```
print("This is my first Python program.")
```

Calling the **`print`** *method*

statements

Character Data

- Our first program is printing a string of characters.
- We are usually interested in manipulating more than one character at a time, but we treat single characters and strings of characters the same way.
- We can use either ' and " for delimit one or more characters, and we can extend them over more than one line by using "\".
- For now, we use character data for input and output only.

Printing Output

```
print("This is my first Python program.")
```

- **`print`** takes a list of elements in parentheses separated by commas
  - Because the element is a string, it will be printed it as is.
  - After printing, it will move on to a new line of output

A second program

Problem – write a program which can find the average of three numbers.

Let's list the steps that our program must perform to do this:

1. Add up these values
2. Divide the sum by the number of values
3. Print the result

Each of these steps will be a different statement.

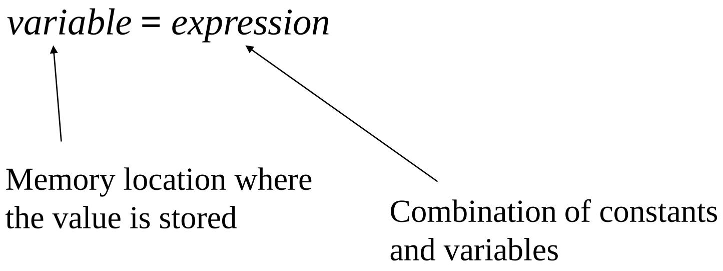
Writing Our Second Program

1. Add up these values → `sum = 2 + 4 + 6`
2. Divide the sum by the number of values
3. Print the result

`sum = 2 + 4 + 6` ← an assignment statement

Assignment Statements

- Assignment statements take the form:



variable = *expression*

Memory location where the value is stored

Combination of constants and variables

Expressions

- Expressions combine values using one of several ***operations***.
- The operations being used is indicated by the ***operator***:

+	Addition
-	Subtraction
*	Multiplication
/	Division

Expressions – Some Examples

2 + 5

4 * value

x / y

What Can Go On The Left-Hand Side?

- There are limits therefore as to what can go on the left-hand side of an assignment statement.
- The left-hand side must indicate a name with which a value can be associated
- This name must follow the naming rules

Python “Types”

- Integers: **`5`**
- Floats: **`1.2`**
- Booleans: **`true`**
- Strings: **`"anything"`** or **`'something'`**
- Lists: **`[,]   [‘a’,1,1.3]`**
- Others we will see

What Is A Type?

- A type in Python essentially defines two things:
  - The internal structure of the type (what is contains)
  - The kinds of operations you can perform
- **`'abc'.capitalize()`** is a method you can call on strings, but not integers
- Some types have multiple elements (collections), we’ll see those later

Fundamental Types

- Integer
  - -1, -27, (to ±2^{32}-1)
  - -127L – L suffix mean any length, but potentially very slow
- Floating Point (Real)
  - 3.14, 10., .001, 3.14e-10, 0e0
- Boolean (True or False values)
  - True, False (note the capital letter)

Converting Types

- A character '**`1`**' is not an integer **`1`**. We'll see more on this later, but take my word for it.
- You need to convert the value returned by the **`input`** command (characters) into an integer
- **`int("123")`** yields the integer **`123`**

Type Conversion

- Conversion functions
  - **`int(some_var)`** returns an integer
  - **`float(some_var)`** returns a float
  - **`str(some_var)`** returns a string
- should check out what works:
  - **`int(2.1)`** → **`2`**, **`int('2')`** → **`2`**, **`int('2.1')`** fails
  - **`float(2)`** → **`2.0`**, **`float('2.0')`** → **`2.0`**
  - **`float('2')`** → **`2.0`**, **`float(2.0)`** → **`2.0`**
  - **`str(2)`** → **`'2'`**, **`str(2.0)`** → **`'2.0'`**, **`str('a')`** → **`'a'`**

Two Types of Division

- The standard division operator (/) yields a floating point result no matter the type of its operands:
  - `2/3` → **`0.6666666666666666`**
  - `4.0/2` → **`2.0`**
- Integer division (//) yields only the integer part of the divide (its type depends on its operands):
  - `2//3` → **`0`**
  - `4.0//2` → **`2.0`**

Modulus Operator

- The modulus operator (%) give the integer remainder of division:
  - `5 % 3` → `2`
  - `7.0 %` 3 →`1.0`
- Again, the type of the result depends on the type of the operands.

Order of Operations and Parentheses

Operator	**Description**
`()`	Parentheses (grouping)
`**`	Exponentiation
`+x`, `-x`	Positive, Negative
`*`, `/`, `%`, `//`	Multiplication, Division, Remainder, Quotient
`+`, `-`	Addition, Substraction

Precedence of *,/ over +,–is the same, but there precedents for other operators as well
Remember, parentheses always takes precedence

Writing Our Second Program

- **`sum = 2 + 4 + 6;`**
- Divide the sum by the number of values
- Print the result

`average = sum / 3;`

Names that describe what the values represent

Writing Our Second Program

1. **`sum = 2 + 4 + 6`**
2. **`average = sum / 3;`**
3. Print the result

```
print("The average is ", average)
```

variable name

Writing Our Second Program

```
sum = 2 + 4 + 6
average = sum / 3;
print("The average is ", average)
```

Save as a “Module”

- When you save a file, such as our first program, and place a **`.py`** suffix on it, it becomes a python module
- You run the module from the IDLE menu to see the results of the operation
- A module is just a file of python commands

Errors

- If there are interpreter errors, that is Python cannot run your code because the code is somehow malformed, you get an error
- You can them import the program again until there are no errors

Common Error

- Using IDLE, if you save the file without a `.py` suffix, it will stop colorizing and formatting the file.
- Resave with the .py, everything is fine

Variables and Identifiers

- Variables have names – we call these names ***identifiers***.
- Identifiers identify various elements of a program (so far the only such element are the variables.
- Some identifiers are standard.

Identifier Rules

- An identifier must begin with a letter or an underscore **_**
- Java is case sensitive upper case (capital) or lower case letters are considered different characters. `Average`, `average` and `AVERAGE` are three different identifiers.
- Numbers can also appear after the first character.
- Identifiers can be as long as you want but names that are too long usually are too cumbersome.
- Identifiers cannot be reserved words (special words like `int`, `main`, etc.)

Some Illegal Identifiers

Illegal Identifier	Reason	Suggested Identifier
`my age`	Blanks are not allowed	`myAge`
`2times`	Cannot begin with a number	`times2` or `twoTimes`
`four*five`	* is not allowed	`fourTimesFive`
`time&ahalf`	& is not allowed	`timeAndAHalf`

Using Stepwise Refinement to Design a Program

- You should noticed that when we write a program, we start by describing the steps that our program must perform and we subsequently refine this into a long series of more detailed steps until we are writing individual steps. This is called ***stepwise refinement***.
- Stepwise refinement is one of the most basic methods for developing a program.

Another Version of Average

- Let’s rewrite the average program so it can find the average any 3 numbers we try:
- We now need to:
  1. Find our three values
  2. Add the values
  3. Divide the sum by 3
  4. Print the result

Writing Average3b

This first step becomes:

1.1 Find the first value

1.2 Find the second value

1.3 Find the third value

2. Add the values
3. Divide the sum by 3
4. Print the result

Reading from the keyboard

- The function

```
value1 = input("What is the first value?")
```

- prints “Give me a value” on the python screen and waits till the user types something (anything), ending with Enter
- Warning, it returns a string (sequence of characters), no matter what is given, even a number ('1' is not the same as 1, different types)
- We can fix this by adding the program

```
value1 = int(value1)
```

Writing the input statements in Average3b

We can read in a value by writing:

```
value1 = input("What is the first value?")
value1 = int(value1)

value2 = input("What is the second value?")
value2 = int(value2)

value3 = input("What is the third value?")
value3 = int(value3)
```

2. Add the values
3. Divide the sum by 3
4. Print the result

Writing the assignments statements in Average3b

```
value1 = input("What is the first value?")
value1 = int(value1)

value2 = input("What is the second value?")
value2 = int(value2)

value3 = input("What is the third value?")
value3 = int(value3)
sum = value1 + value2 + value3
```

3. Divide the sum by 3
4. Print the result

Adding up the three values

Writing the assignments statements in Average3b

```
     value1 = input("What is the first value?")
value1 = int(value1)

value2 = input("What is the second value?")
value2 = int(value2)

value3 = input("What is the third value?")
value3 = int(value3)
sum = value1 + value2 + value3
average = sum / 3
```

4. Print the result

Calculating the average

`Average3b.py`

```
value1 = input("What is the first value?")
value1 = int(value1)

value2 = input("What is the second value?")
value2 = int(value2)

value3 = input("What is the third value?")
value3 = int(value3)

sum = value1 + value2 + value3
average = sum / 3
print("The average is ", average)
```

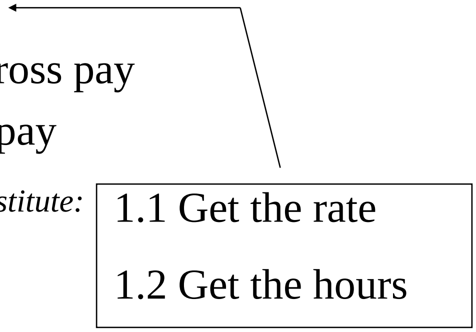
Another example – calculating a payroll

- We are going to write a program which calculates the gross pay for someone earning an hourly wage.
- We need two pieces of information:
  - the hourly rate of pay
  - the number of hours worked.
- We are expected to produce one output: the gross pay, which we can find by calculating:
  - Gross pay = Rate of pay * Hours Worked

Our Design for payroll

1. Get the inputs
2. Calculate the gross pay
3. Print the gross pay

We can substitute:



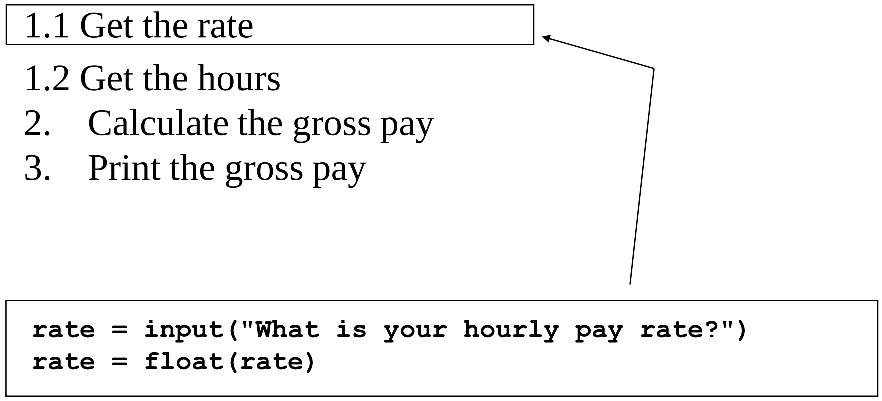
1.1 Get the rate

1.2 Get the hours

Coding the payroll program

- Before we code the payroll program, we recognize that the values (**`rate`**, **`hours`** and **`gross`**) may ***not*** necessarily be integers.
- We will convert the inputted values to **`float`** values.

Developing The Payroll Program (continued)



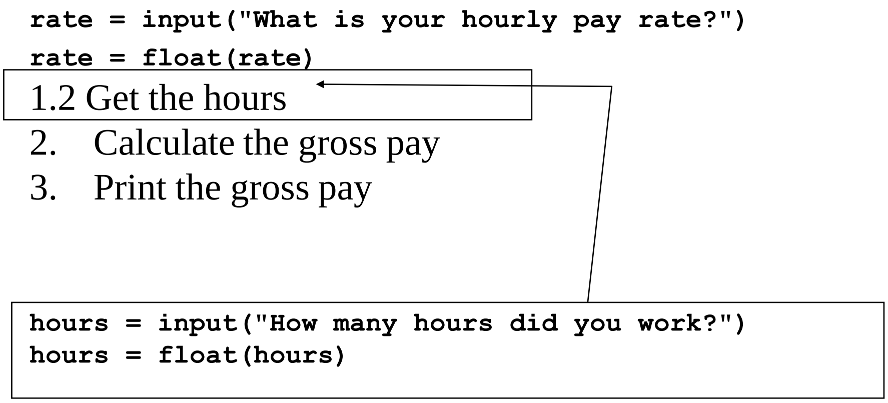
1.1 Get the rate

1.2 Get the hours

2. Calculate the gross pay
3. Print the gross pay

```
rate = input("What is your hourly pay rate?")
rate = float(rate)
```

Developing The Payroll Program (continued)



```
rate = input("What is your hourly pay rate?")
rate = float(rate)
```

1.2 Get the hours

2. Calculate the gross pay
3. Print the gross pay

```
hours = input("How many hours did you work?")
hours = float(hours)
```

Developing The Payroll Program (continued)

```
rate = input("What is your hourly pay rate?")
rate = float(rate)
hours = input("How many hours did you work?")
hours = float(hours)
```

2. Calculate the gross pay
3. Print the gross pay

```
gross = rate * hours
```

Developing The Payroll Program (continued)

```
rate = input("What is your hourly pay rate?")
rate = float(rate)
hours = input("How many hours did you work?")
hours = float(hours)
gross = rate * hours
```

3. Print the gross pay

```
print("Your gross pay is $", gross)
```

Payroll.py

```
rate = input("What is your hourly pay rate?")
rate = float(rate)

hours = input("How many hours did you work?")
hours = float(hours)

gross = rate * hours;
print("Your gross pay is $", gross
```

Comments

- Our program is a bit longer than our previous programs and if we did not know how to calculate gross pay, we might not be able to determine this from the program alone.
- It is helpful as programs get much longer to be able to insert text that explains how the program works. These are called ***comments***. Comments are meant for the human reader, not for the computer.
- A comment begins with a # (pound sign)
- This means that from the # to the end of that line, nothing will be interpreted by Python.
- You can write information that will help the reader with the code

```
# This program calculates the gross pay for an
# hourly worker
# Inputs - hourly rate and hours worked
# Output - Gross pay

# Get the hourly rate
rate = input("What is your hourly pay rate?")
rate = float(rate)

# Get the hours worked
hours = input("How many hours did you work?")
hours = float(hours)

# Calculate and display the gross pay
gross = rate * hours
print("Your gross pay is $", gross)
```

Example – A program to convert pounds to kilograms

- Our program will convert a weight expressed in pounds into kilograms.
  - Our input is the weight in pounds.
  - Our output is the weight in kilograms
  - We also know that

    Kilograms = Pounds / 2.2

Pounds to Kilograms Program (continued)

- Our program must:
  1. Get the weight in pounds
  2. Calculate the weight in kilograms
  3. Print the weight in kilograms

Pounds to Kilograms Program (continued)

- Our program must:
  1. Get the weight in pounds
  2. Calculate the weight in kilograms
  3. Print the weight in kilograms

```
lbs = input("What is the weight in pounds?")
lbs = float(lbs)
```

Pounds to Kilograms Program (continued)

```
lbs = input("What is the weight in pounds?")
lbs = float(lbs)
```

2. Calculate the weight in kilograms
3. Print the weight in kilograms

```
kg = lbs / 2.2
```

Pounds to Kilograms Program (continued)

```
lbs = input("What is the weight in pounds?")
lbs = float(lbs)
kg = lbs / 2.2
```

3. Print the weight in kilograms

```
print("The weight is ", kg, " kilograms")
```

ConvPounds.py

```
# Convert pounds to kilograms
# Input - weight in pounds
# Output - weight in kilograms

# Get the weight in pounds
lbs = input("What is the weight in pounds?")
lbs = float(lbs)

# Calculate and display the weight in kilograms
kg = lbs / 2.2;
print("The weight is ", kg, " kilograms")
```

Another Example – The Area and Circumference of A Circle

- Our program will calculate the area of a rectangle.
  - Our input is the length and width.
  - Our output is the area.
  - We also know that

    $\text{Circumference} = 2 \times \pi \times \text{Radius}$

    $\text{Area} = \pi \times \text{Radius}^2$

Our Program’s Steps

1. Get the radius
2. Calculate the circumference
3. Calculate the area
4. Print the circumference and the area

Our Program’s Steps (continued)

1. Get the radius
2. Calculate the circumference
3. Calculate the area
4. Print the circumference and the area

```
radius_str = input("Enter the radius of your circle:")
radius_int = int(radius_str)
```

Our Program’s Steps (continued)

```
radius_str = input("Enter the radius of your circle:")
radius_int = int(radius_str)
```

2. Calculate the circumference
3. Calculate the area
4. Print the circumference and the area

```
circumference = 2 * math.pi * radius_int
```

The math package includes the value of π

Our Program’s Steps (continued)

```
radius_str = input("Enter the radius of your circle:")
radius_int = int(radius_str)

circumference = 2 * math.pi * radius_int
```

3. Calculate the area
4. Print the circumference and the area

```
area = math.pi * radius_int * radius_int
```

Our Program’s Steps (continued)

```
radius_str = input("Enter the radius of your circle:")
radius_int = int(radius_str)

circumference = 2 * math.pi * radius_int
area = math.pi * radius_int * radius_int
```

4. Print the circumference and the area

```
print ("The circumference is: ", circumference, \
       " and the area is: ", area)
```

```
radius_str = input("Enter the radius of your circle:")
radius_int = int(radius_str)

circumference = 2 * math.pi * radius_int
area = math.pi * radius_int * radius_int

print ("The circumference is: ", circumference, \
       " and the area is: ", area)
```

We need to import the math package; that requires
our program to have `import math` *at the top of the program*

We also need to include comments

```
# Calculate the area and circumference of a circle
# from its radius
import math

# Get the radius
radius_str = input("Enter the radius of your circle:")
radius_int = int(radius_str)

#Calculate the circumference
circumference = 2 * math.pi * radius_int

# Calculate the area
area = math.pi * radius_int * radius_int

# Print the circumference and the area
print ("The circumference is: ", circumference, \
       " and the area is: ", area)
```

The Rules

1. Think before you program
2. A program is a human-readable essay on problem solving that also happens to execute on a computer.
3. The best way to improve your programming and problem solving skills is to practice.
4. A foolish consistency is the hobgoblin of little minds
5. Test your code, often and thoroughly!