Introduction to Python

Systems Programming
COMP-6060
Mansur Mulk, Ph.D.

FANSHAWE

What is a Computer

- A computer is a digital electronic machine that can be programmed to achieve different tasks (source)
- Computers contain integrated circuits made up of millions of nano-sized transistors that act as electronic logic gates (switches)
- These logic gates make up complex integrated circuits, that allow for computers to be programmable

Computer Programs

A computer program tells a computer the sequence of steps needed to complete a specific task The program consists of a very large number of primitive (simple) instructions.

Computers can carry out a wide range of tasks because they can execute different programs Each program is designed to direct the computer to work on a specific task

Programming Languages

- Since computers are made up of logic circuits, the only 'language' they understand is Boolean logic (ones and zeros)
- In order to program computers, we would need to provide instructions in machine code
- This is incredibly hard and error prone, since binary is not how we normally think

Programming Languages

- This is where programming languages come in…
- Programming languages provide a notation system to write computer programs in text form, which then gets translated into machine code (binary)
- There are hundreds of programming languages that have been developed, and in this course we will be focusing on Pythonprogramming language

Programming Languages

- Hardware & Software
- Algorithms
- Python & Programming Environments
- Types of Error in Programs: Syntax & Logic Errors

Hardware

Hardware consists of the physical elements in a computer system. Some very visible examples are the monitor, the mouse, external storage, and the keyboard.

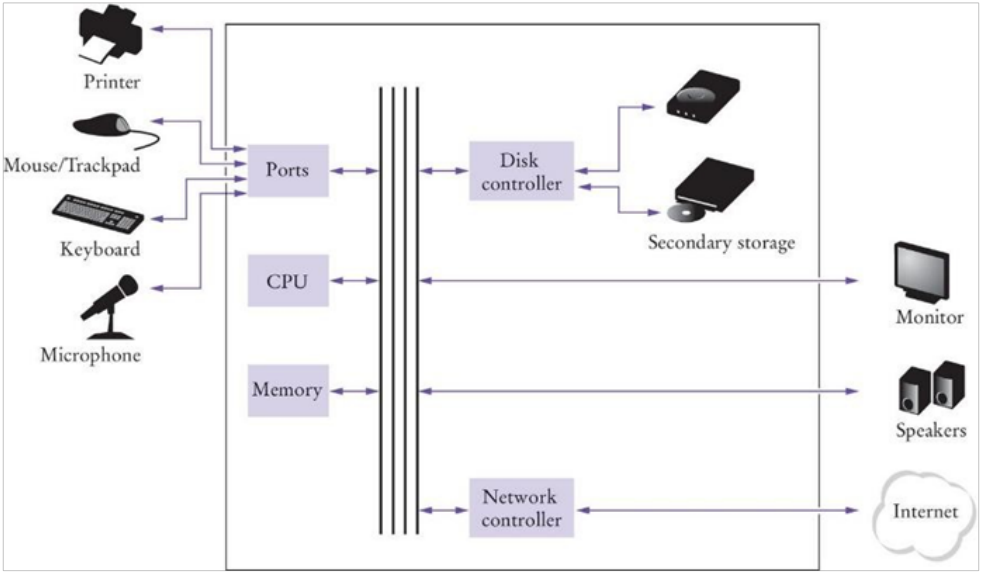
The central processing unit (CPU) performs program control and data processing. Storage devices include memory (RAM) and secondary storage:

- Hard disk
- Flash drives
- CD/DVD drives

Input / output devices allow the user to interact with the computer:

- Mouse, keyboard, printer, screen…

Simple view of Computer Component



Introduction to Algorithms

If you want a computer to perform a task, you start by writing an algorithm

An Algorithm is:

- a sequence (the order mattering) of actions to take to accomplish the given task
- An algorithm is like a recipe; it is a set of instructions written in a sequence that achieves a goal

For complex problems software developers write an algorithm before they attempt to write a computer program

Developing algorithms is a fundamental problem solving skill

- It has uses in many fields outside of Computer Science

Algorithms: Formal Definitions

An <u>algorithm</u> describes a sequence of steps that is:

1. **Unambiguous**
   a. No "assumptions" are required to execute the algorithm
   b. The algorithm uses precise instructions

2. **Executable**
   a. The algorithm can be carried out in practice

3. **Terminating**
   a. The algorithm will eventually come to an end, or halt

Problem Solving: Algorithm Design

Algorithms are simply plans

- Detailed plans that describe the steps to solve a specific problem

You already know quite a few

- Calculate the area of a circle
- Find the length of the hypotenuse of a triangle

Some problems are more complex and require more steps

- Calculate PI to 100 decimal places
- Calculate the trajectory of a missile

Bank Account Example

Problem Statement:
You put $10,000 into a bank account that earns 5 percent interest per year. How many years does it take for the account balance to be double the original?

How would you solve it?

- Manual method
- Make a table
- Add lines until done
- Use a spreadsheet!
- Write a formula
- Per line, based on the line above

year	balance
0	10000
1	10000.00 x 1.05 = 10500.00
2	10500.00 x 1.05 = 11025.00
3	11025.00 x 1.05 = 11576.25
4	11576.25 x 1.05 = 12155.06

Translate to Pseudocode

You put $10,000 into a bank account that earns 5 percent interest per year. How many years does it take for the account balance to be double the original?

Break it into steps

- Start with a year value of 0 and a balance of $10,000
- Repeat the following while the balance is less than $20,000
  - Add 1 to the year value
  - Multiply the balance by 1.05
    - (5% increase)

year	balance
0	10000

year	balance
0	10000
1	10500
14	19799.32
15	20789.28

<u>Report the final year value as the answer</u>

Translate to Pseudocode

Pseudocode

- Half-way between natural language and a programming language

Modified Steps:

- Set the year value of 0
- Set the balance to $10,000
- While the balance is less than $20,000
  - Add 1 to the year value
  - Multiply the balance by 1.05
- Report the final year value as the answer

<u>The pseudocode is easily translated into Python</u>

C

FANSHAWE

PYTHON

Python is a high-level, easy, interpreted, general-purpose, and dynamic programming language. It supports object-oriented programming approach. It is straight forward to learn, and its elegant syntax allows programmers to express concepts in fewer lines of code as compared to other languages such as C, C++, or Java.

PYTHON

- It has excellent data-structure, which makes it a unique language for scripting and application development on most platforms.
- Python is a platform-independent language, which means a Python program can run on any operating system like Windows, Macintosh, Linux, etc.
- Python runs on an interpreted system, which means that code can be executed line by line.
- Python is a dynamically typed, so we don't need to declare the data type of variable such as x=20 assigns an integer value to an integer variable.

The Python Language

- In the early 1990's, Guido van Rossum designed what would become the Python programming language
- Van Rossum was dissatisfied with the languages available
  - They were optimized to write large programs that executed quickly
- He needed a language that could not only be used to create programs quickly but also make them easy to modify
  - It was designed to have a much simpler and cleaner syntax than other popular languages such as Java, C and C++ (making it easier to learn)
  - Python is interpreted, making it easier to develop and test short programs
- Python programs are executed by the Python interpreter
  - The interpreter reads your program and executes it

Programming Environment

There are several ways of creating a computer program

- Using an Integrated Development Environment (IDE)
- Using a text editor

IDE vs. Interpreter

- Python is the Interpreter
- PyCharm is the IDE

You should use the method you are most comfortable with.

- In this class, I will use the PyCharm Educational Version

What is an IDE

- IDE is an abbreviation for Integrated Development Environment
- An IDE is a software the provides capabilities to develop, run, and debug software.
- For this class, we will be using Visual Studio Code as our IDE

Introduction to Python

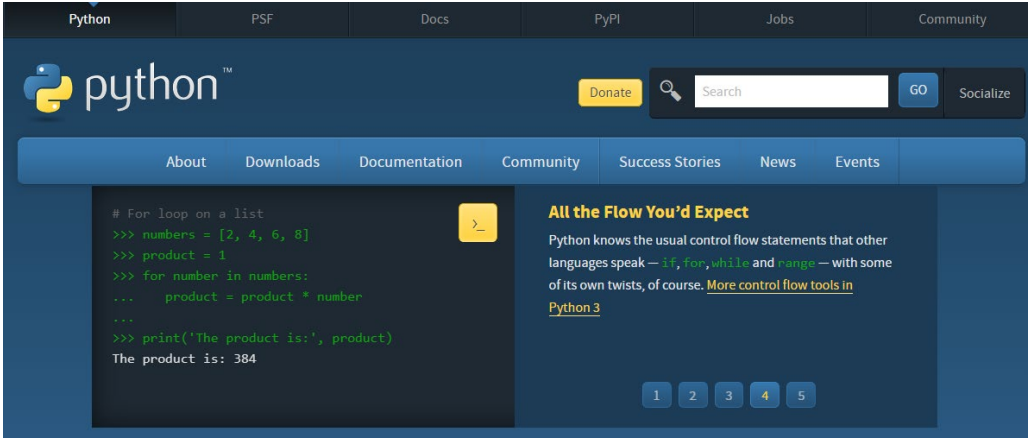
- In order to write Python code, we need to create .py files (similar to .c and .h files for C language)

- Python code then gets interpreted into machine code, which means we need to install the Python interpreter

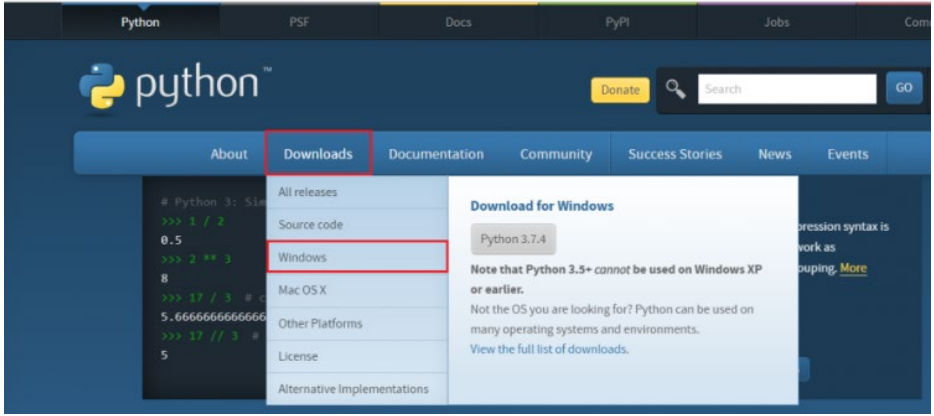
Python Setup Guide

Step-1 Set Up for Download

- Python can be downloaded from the Python Software Foundation website at https://www.python.org/



Python Setup Guide (Windows)



Python Setup Guide (Windows)

Step-2 Select Installer:

Select a link either **window x86-64executable installer** to download Python for 64-bit OS or **window x86 executable installer** for 32-bit OS.

Python Setup Guide (Windows)

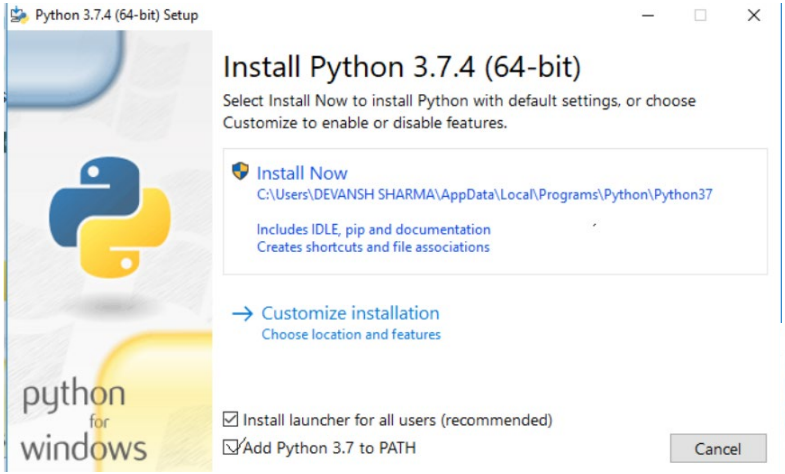
Step-3 Run the installer

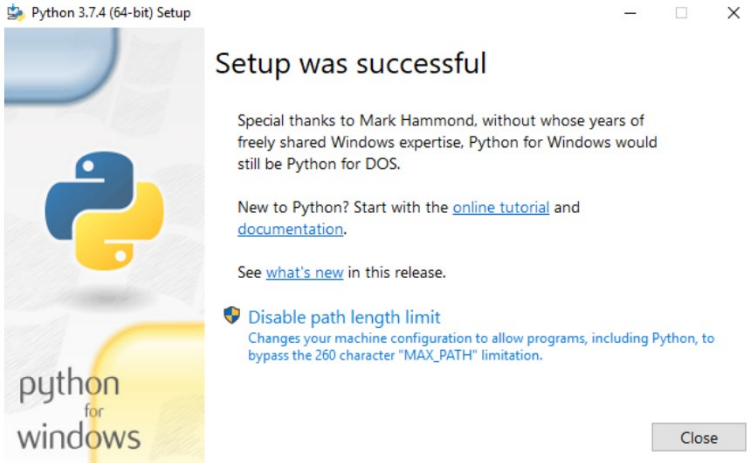
Once you download the installer, run that installer.

1.Select the install launcher for **all user** and **Add Python 3.7 to Path checkbox.** Older versions of Python do not support the **Add Python to Path** checkbox.

2. Click on **install now** option, which is the recommended installation options.

Python Setup Guide (Windows)

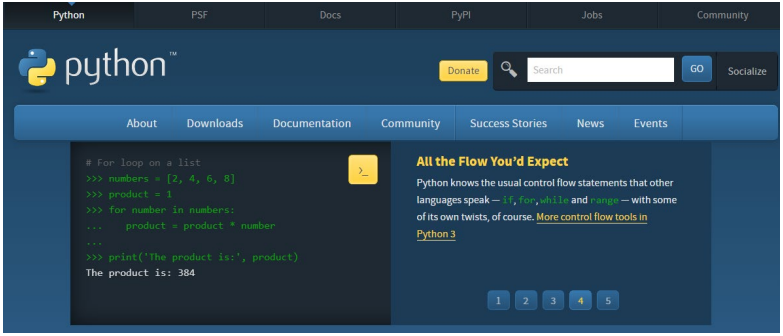




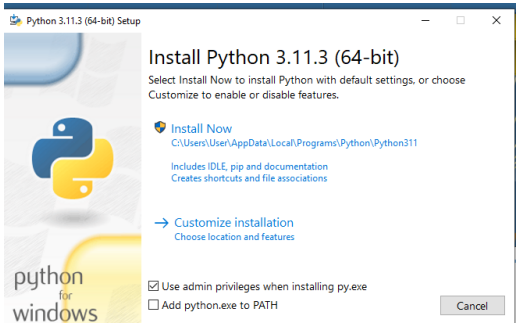
https://www.tutorialandexample.com/python-setup-guide-and-installation

Python Setup Guide(MAC)

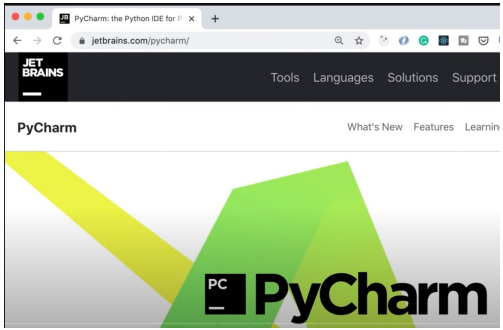
- Python can be downloaded from the Python Software Foundation website at https://www.python.org/

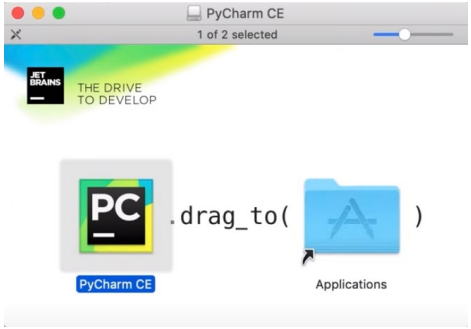


Python Setup Guide(MAC)



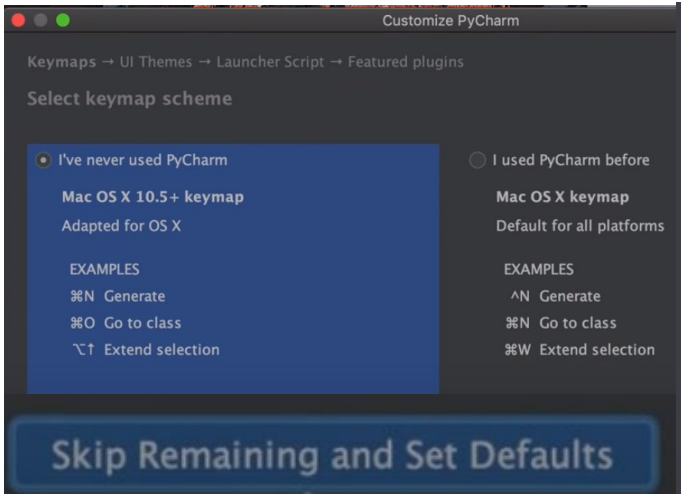
Code Editor

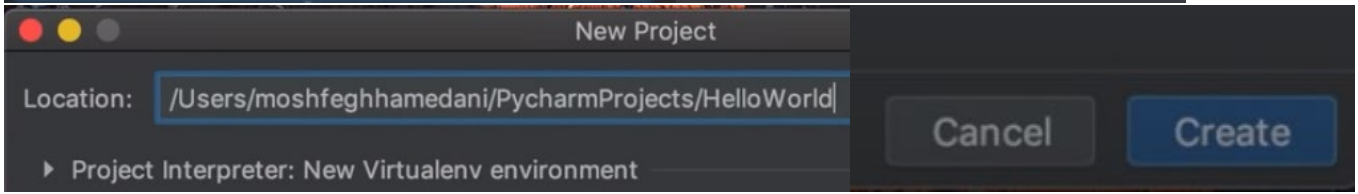




Python Setup Guide(MAC)

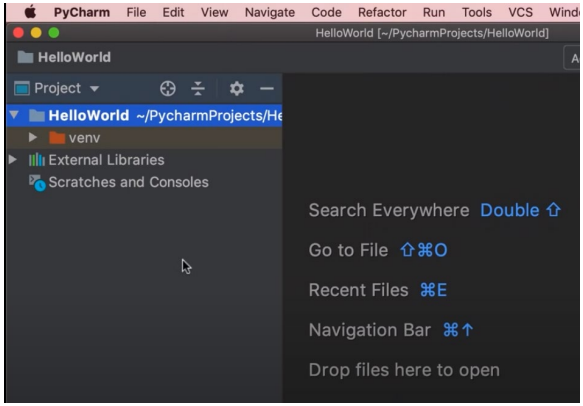
Code Editor

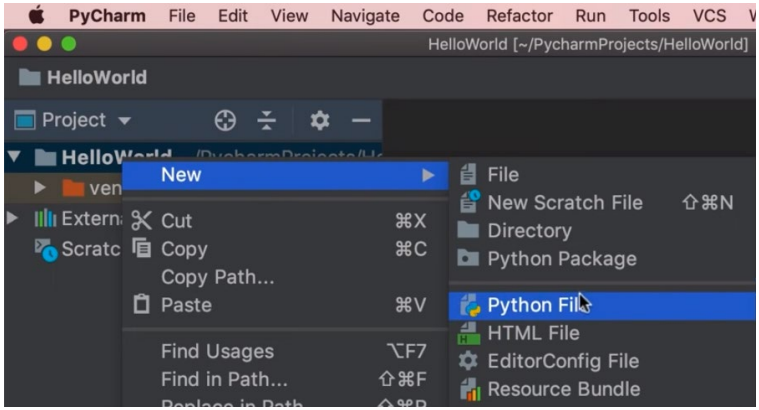


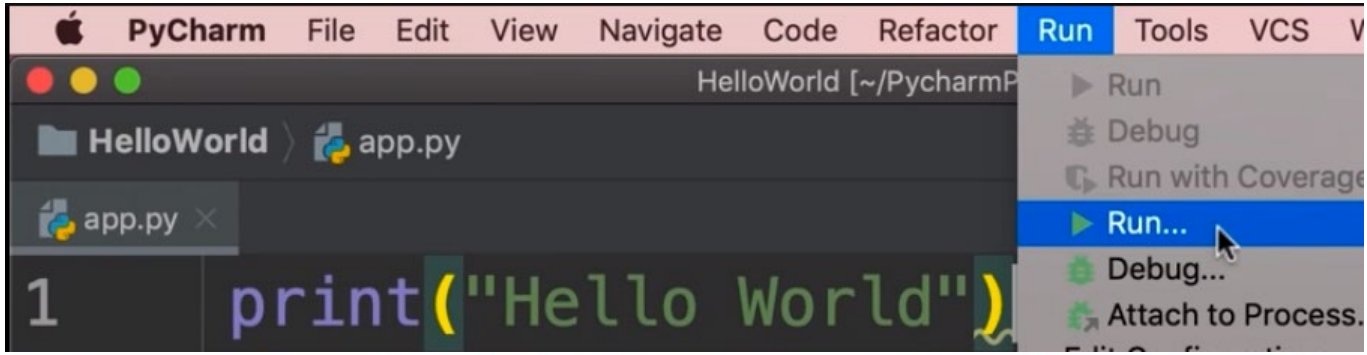


Python Setup Guide(MAC)

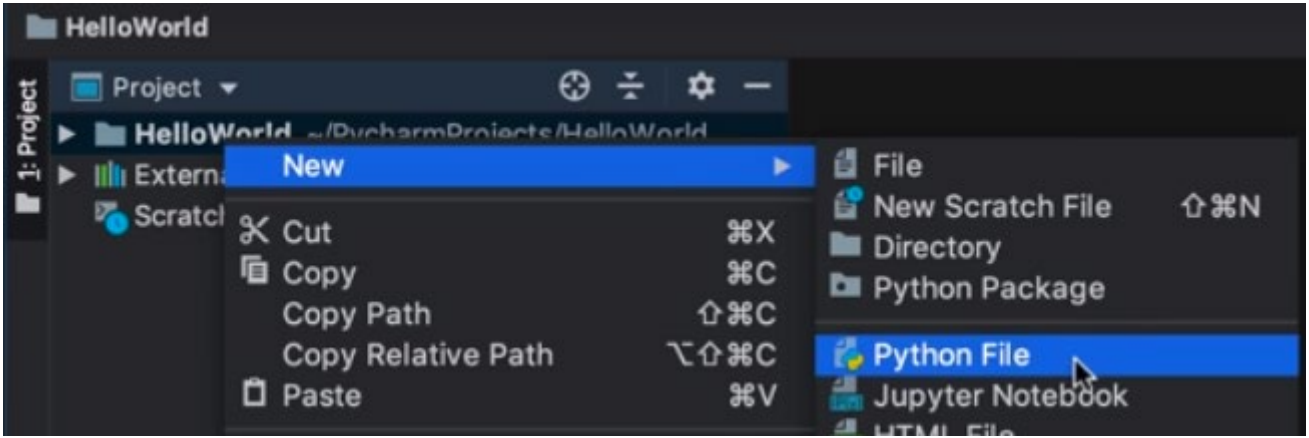
Code Editor

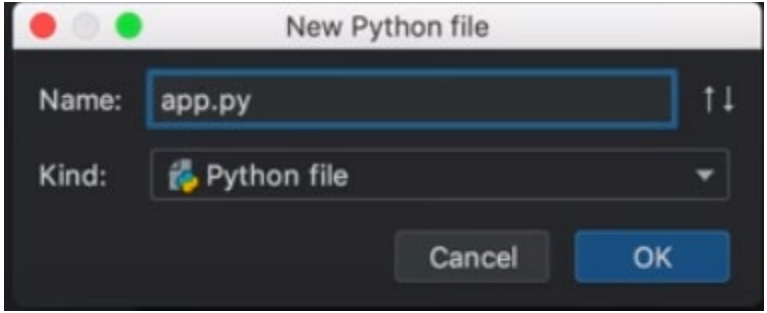






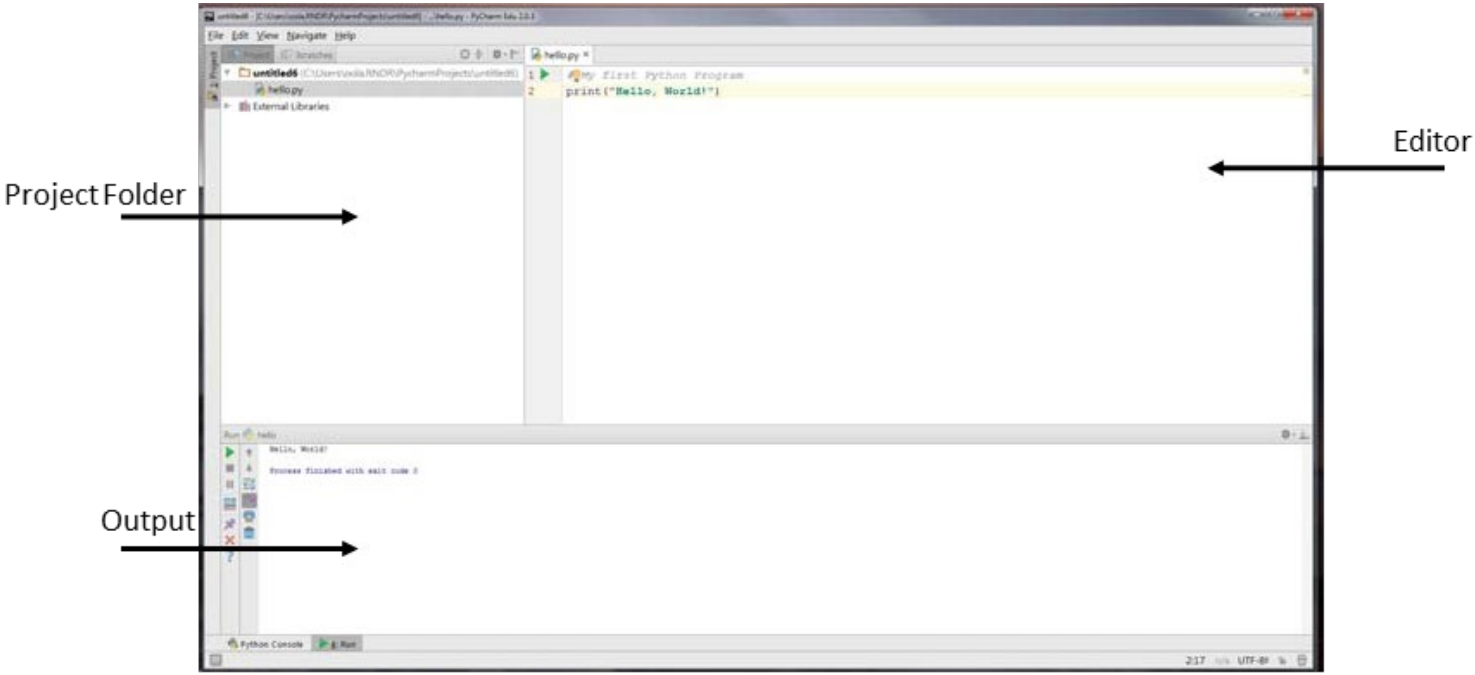
Let's write some code!





FANSHAWE

PyCharm IDE



Your First Program

Traditional 'Hello World' program in Python

We will examine this program in the next section:

- Typing the program into your IDE would be good practice!
- Be careful of spelling e.g., 'print' vs. 'primt'
- PyTHon iS CaSe SeNsItiVe.

>Python & Programming Environments

```
# My first Python program.
print("Hello, World!")
```

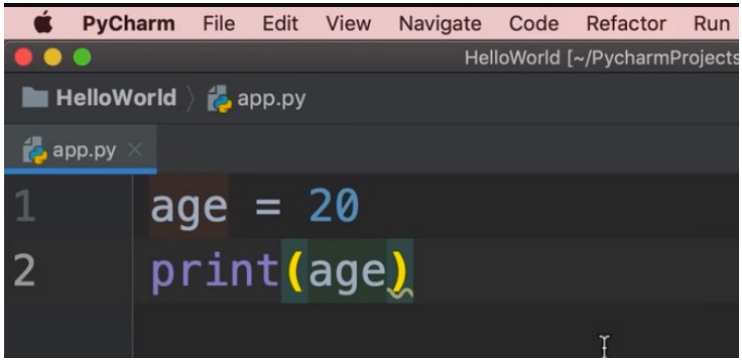
Analyzing Your First Program

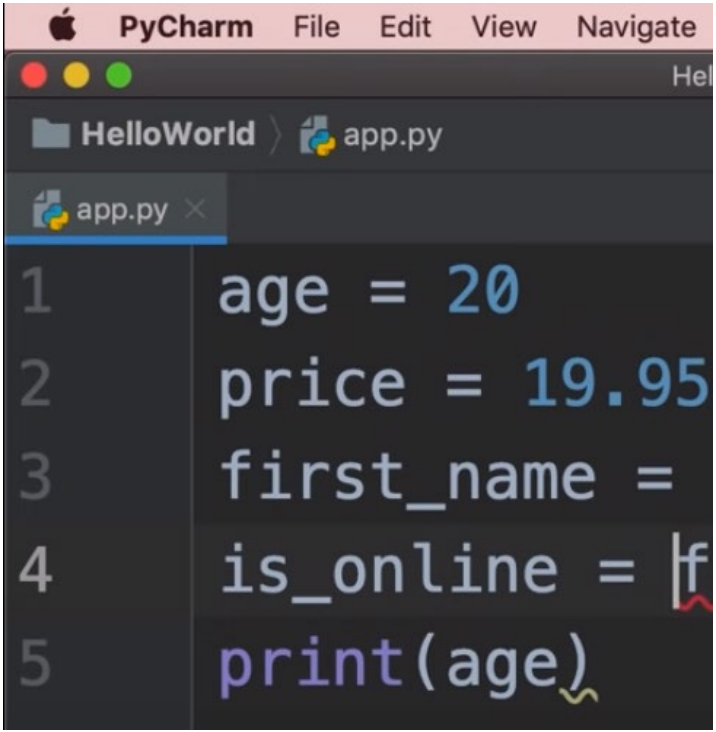
- A Python program contains one or more lines of instructions (statements) that will be translated and executed by the interpreter

```
# My first Python program
Print("Hello World!")
```

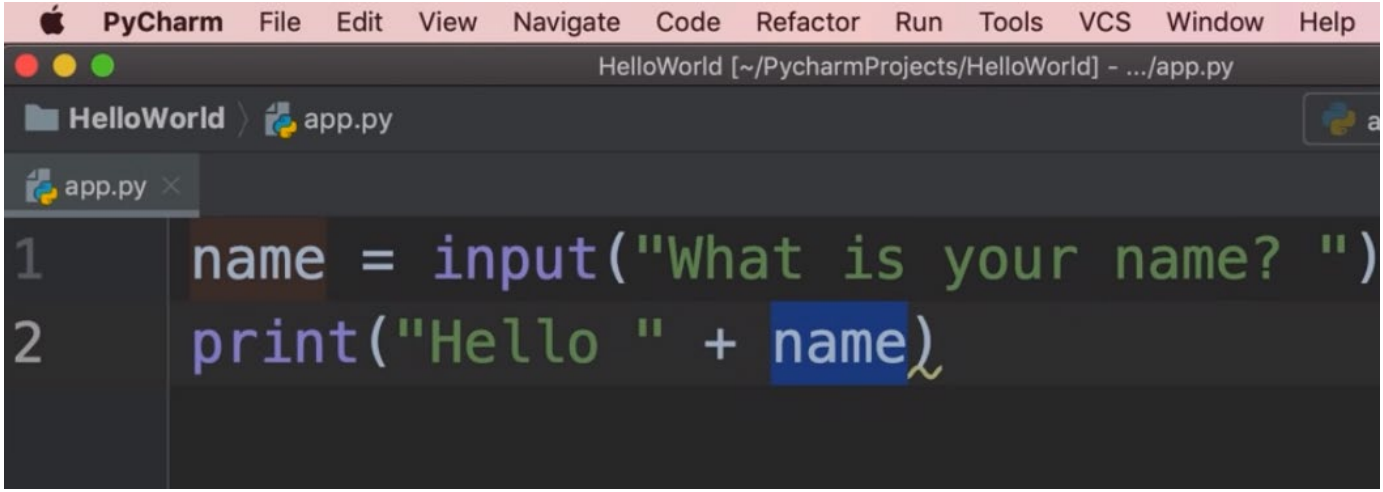
- The first line is a comment (a statement that provides descriptive information about the program to programmers).
- The second line contains a statement that prints a line of text onscreen "Hello, World!"

Variable


PyCharm File Edit View Navigate Code Refactor Run
HelloWorld [~/PycharmProjects
HelloWorld app.py
app.py
1 age = 20
2 print(age)


PyCharm File Edit View Navigate
HelloWorld app.py
app.py
1 age = 20
2 price = 19.95
3 first_name =
4 is_online = f
5 print(age)

Inputs From User



String concatenation

Data Type

- Numbers
- String
- Boolean

```python
birth_year = input("Enter your birth year: ")
age = 2020 - int(birth_year)
print(age)
```

```python
first = input("First: ")
second = input("Second: ")
sum = int(first) + int(second)
print(sum)
```

```python
first = input("First: ")
second = input("Second: ")
sum = float(first) + float(second)
print("Sum: " + str(sum))
```

```python
course = 'Python for Beginners'
print(course.upper())
```

Data Type

- Numbers
- String
- Boolean

```python
birth_year = input("Enter your birth year: ")
age = 2020 - int(birth_year)
print(age)
```

```python
first = input("First: ")
second = input("Second: ")
sum = int(first) + int(second)
print(sum)
```

```python
first = input("First: ")
second = input("Second: ")
sum = float(first) + float(second)
print("Sum: " + str(sum))
```

```python
course = 'Python for Beginners'
print(course.upper())
```

Data Type

- Numbers
- String
- Boolean

HelloWorld [~/PycharmProjects/HelloWorld] - .../app.py

HelloWorld 〉 app.py

app.py

```
course = 'Python for Beginners'
print(course.find('|'))
print(course)
```

HelloWorld 〉 app.py

app.py

```
course = 'Python for Beginners'
print(course.replace('x', '4'))
```

HelloWorld 〉 app.py

app.py

```
course = 'Python for Beginners'
print('Python' in course)
```

HelloWorld 〉 app.py

app.py

```
x = (10 + 3) * 2
```

HelloWorld 〉 app.py

app.py

```
x = 3 == 2
print(x)
```

Mathematical and Logical Operations

- Addition, subtraction, multiplication
  - 2 + 2, 42 – 6, 4 * 3
- Division (type matters)
  - 3 / 2 = ?
- Modulus
  - 3 % 2 = 1
- Exponentiation
  - 3 ** 2 = 9
- Logical operators
  - and, or, not
  - ==, !=, <, <=, >, >=

print Statement

- Places output on the screen

```
i = 34
print i
```

- Use comma between variables you want printed on the same line (i.e., comma will suppress a newline)

```
print i
print i,j
```

- Useful for debugging!

FANSHAWE

Strings

- To create a string:

```
strvar1 = 'abc'

strvar2 = str(123) # can cast objects as strings
```

```
strvar5 = ''
strvar5 += 'cr'
strvar5 += 'ude' # concatenation
```

- String formatting

```
# using string formatting
strvar3 = 'Pi is about %.4f' % 3.142951
 → 'Pi is about 3.1430'

# more formatted strings
strvar4 = '%s Student #%d!' % ('Hello',42)
 → 'Hello Student #42!'
```

FANSHAWE

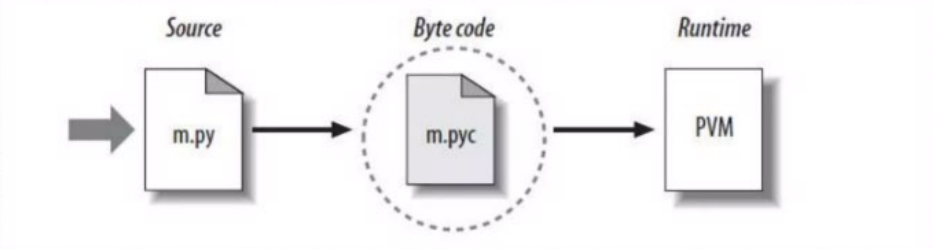
String Operations

```
'hello' + 'world'   →  'helloworld'        # concatenation
'hello' * 3         →  'hellohellohello'   # repetition

'hello'[::-1]       →  'olleh'             # reversing by slice
len('hello')        →     5                # size

'hello' < 'jello'        →  1              # comparison
'e' in 'hello'           → True            # membership

'hello'.find('lo')        →    3           # finding substrings
'hello_world'.count('o')  →    2           # counting substrings

# splitting strings
'hello_world'.split('_')    →  ['hello','world']

# remove whitespace
'hello_world   \n'.strip()  →   'hello_world'
```

FANSHAWE

Python Code Execution

- Python's traditional runtime execution model: source code you type is translated to byte code, which is then run by the Python Virtual Machine. Your code is automatically compiled, but then it is interpreted.



Source code extension is **.py**
Byte code extension is **.pyc** (compiled python code)

FANSHAWE

Error

There are two Categories of Errors:

- Compile-time Errors (aka Syntax Errors)
  - Spelling, capitalization, punctuation
  - Ordering of statements, matching of parenthesis, quotes…
  - No executable program is created by the compiler
  - Correct first error listed, then compile again.
    - Repeat until all errors are fixed

- Run-time Errors (aka Logic Errors)
  - The program runs, but produces unintended results
  - The program may 'crash'

FANSHAWE

Syntax Error

Syntax error are caught by the compiler

What happens if you:

- Miss-capitalize a word:
- Leave out quotes
- Mismatch quotes
- Don't match brackets

```
Print("Hello World!")
print(Hello World!)
print("Hello World!')
print('Hello'
```

Type each example above in PyCharm

What error messages are generated?

FANSHAWE

Logic Error

What happens if you
- Divide by zero
- Misspell output
- Forget to output

```
print(1/0)
print("Hello, Word!")
Remove line 2
```

Programs will compile and run
- The output may not be as expected

Type each example above in PyCharm What

error messages are generated?

FANSHAWE

Summary: Errors and Pseudocode

- A compile-time error is a violation of the programming language rules that is detected by the compiler.
- A run-time error causes a program to take an action that the programmer did not intend.
- Pseudo code is an informal description of a sequence of steps for solving a problem.
- An algorithm for solving a problem is a sequence of steps that is unambiguous, executable, and terminating.

More Information

FANSHAWE