A Merry Overview of the Python Programming Language

By Doug Purcell

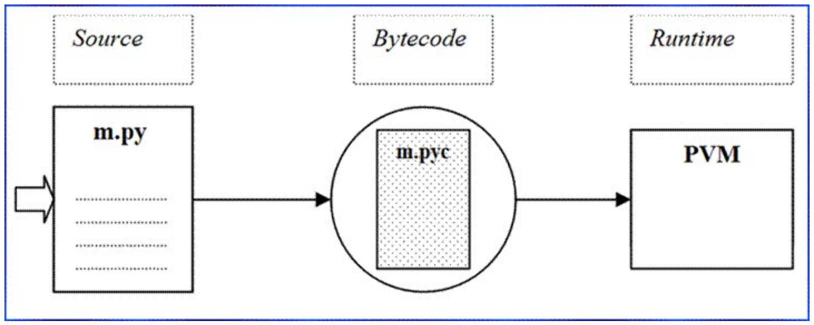
A Very Special Thanks to Our Venue Sponsor!

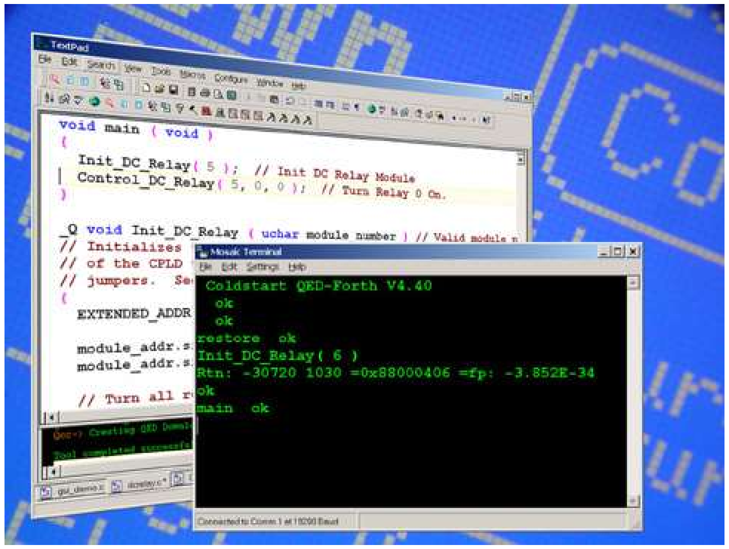
Event venue: www.codemarket.io

Why Learning Python Fundamentals Matter

- Mastering the fundamentals helps you to understand data structures and algorithms better which are a must for technical interviews.
- Strong fundamentals makes learning more advanced topics like web dev, machine learning, and blockchain easier.
- By showing mastery of python it increases your chances of gaining internships and starting a brand new career in the hot software engineering field.

Setting Up Your Python Programming Environment




TextPad
File Edit Search View Tools Macros Configure Window Help
Mosaic Terminal
File Edit Settings Help
Connected to Comm 1 at 19200 Baud

```
void main ( void )
{
  Init_DC_Relay( 5 );  // Init DC Relay Module
  Control_DC_Relay( 5, 0, 0 );  // Turn Relay 0 On.
}

_Q void Init_DC_Relay ( uchar module number ) // Valid module n
// Initializes
// of the CPLD
// jumpers.  Se
{
  EXTENDED_ADDR

  module_addr.s
  module_addr.s

  // Turn all r
```

```
 Coldstart QED-Forth V4.40
  ok
  ok
restore  ok
Init_DC_Relay( 6 )
Rtn: -30720 1030 =0x88000406 =fp: -3.852E-34
ok
main  ok
```

Getting Python Installed On Your Machine

Python is cross-platform so therefore you can install it on multiple machines including Windows, MacOS, or various flavors of Linux.

You Need to Install Python if You Get This Message

```
'python' is not recognized as an
internal or external command,
operable program or batch file
```

Installing Python on Windows

Step one: Download the latest version of python on your machine: `https://www.python.org/downloads`

Step two: Start the Windows installer that matches your system. If you click "Install Now" then Python is installed in the "user" directory, but if you change its location then just remember where it's installed.

Step three: You'll have an option to add Python to PATH which is where the computer searches for Python when you type it via command prompt. If you check this box then Python will be available via this option, if not then an error will occur.

Install Python on macOS

An easy way is to install the latest-and-greatest version of python on macOS is to use the appropriate macOS installer that matches your system: https://www.python.org/downloads

Installing Python on Linux (Ubuntu 18.04)

- Step one: Open up the terminal by typing: ctr + alt + t
- Step two: sudo apt-get update
- Step three: sudo apt-get install python3.7.3

Online Python Interpreters: A Short Term Solution

- Online GDB: https://www.onlinegdb.com/online_python_interpreter
- Repl.it: https://repl.it/languages/python3
- Another online python interpreter: http://mathcs.holycross.edu/~kwalsh/python

Which IDE to Use For Python???

PC

eclipse

PyDev

WING PYTHON IDE

THE INTELLIGENT DEVELOPMENT ENVIRONMENT FOR PYTHON

eric

What I Use

Try Them All Out & See Which Ones *Tick* With You

A comprehensive list of IDEs for Python: **https://wiki.python.org/moin/IntegratedDevelopmentEnvironments**

How to Install Python on All Machines

Windows:

Download the PyCharm installer, run the executable file, and follow the installer steps. Here's the instructions on the PyCharm website: **`https://www.jetbrains.com/help/pycharm/installation-guide.html?section=Windows`**

· **MacOs:**

Download the PyCharm disk image and mount and drag the image to the Applications folder: **`https://www.jetbrains.com/help/pycharm/installation-guide.html?section=macOS`**

· **Linux**: If you have Ubuntu 16.04 you can install PyCharm through the command line using the snappy package manager:

- $ sudo snap apt-get install pycharm-community

Hello World with PyCharm

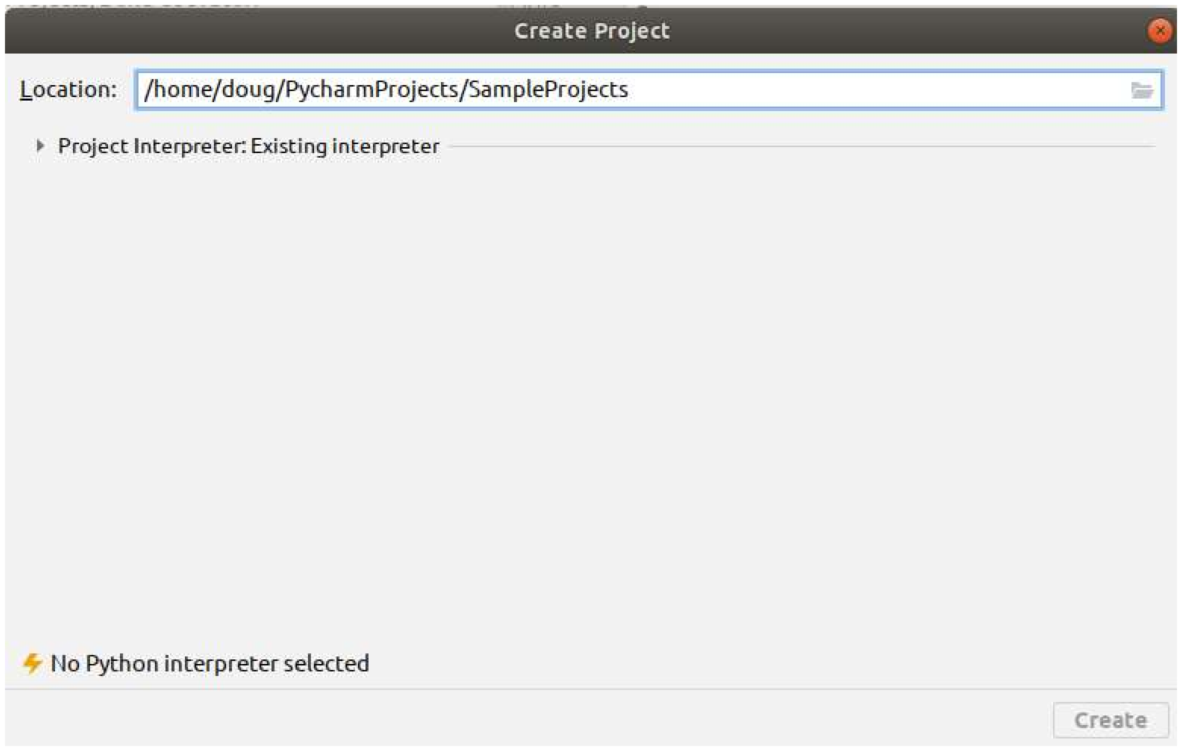
Once PyCharm is installed the next step is to run the proverbial *Hello World* program. Here are the steps...

Create a Brand Spanking New Project

Create a new project by doing the following: *File → New Project*

A project is an organizational unit in PyCharm. Name the project *SampleProjects;*

Here’s a Screenshot of How it Should Look



Location: /home/doug/PycharmProjects/SampleProjects

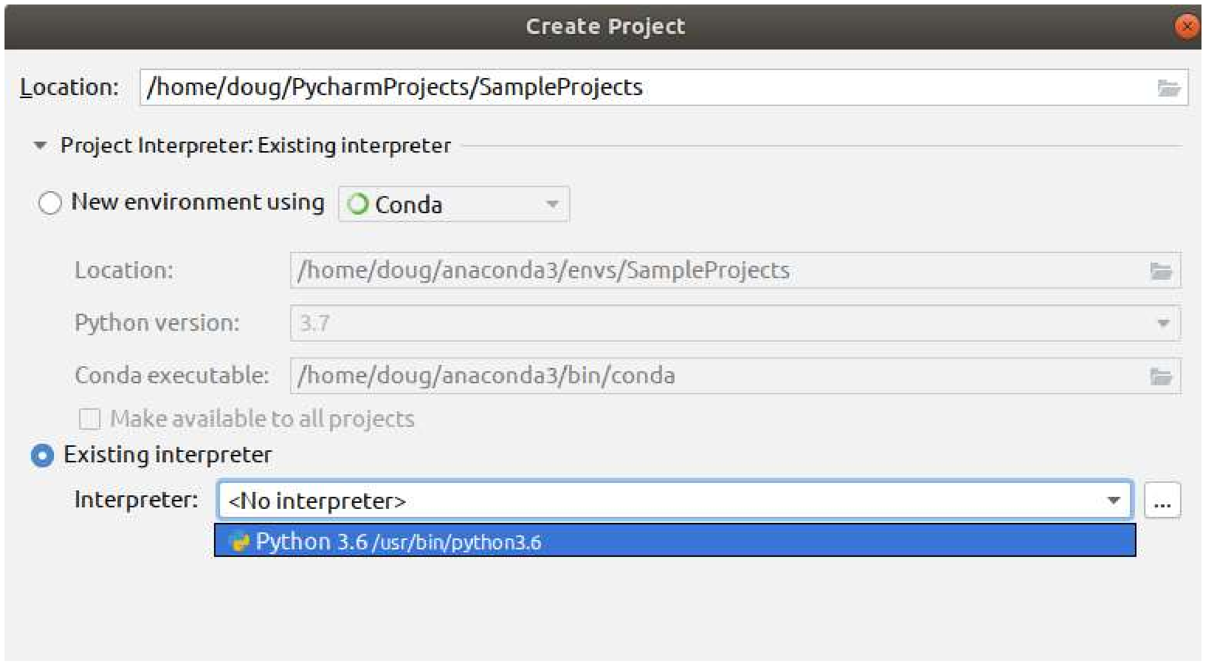
Project Interpreter: Existing interpreter

No Python interpreter selected

Add an Interpreter

Select the python interpreter that you want to use. Click the arrow that’s next to *Project Interpreter:Existing Interpreter,* and then *s*elect a python 3.6 or up.

How Things Should Look?



Create Project

Location: /home/doug/PycharmProjects/SampleProjects

Project Interpreter: Existing interpreter

○ New environment using Conda

Location: /home/doug/anaconda3/envs/SampleProjects

Python version: 3.7

Conda executable: /home/doug/anaconda3/bin/conda

☐ Make available to all projects

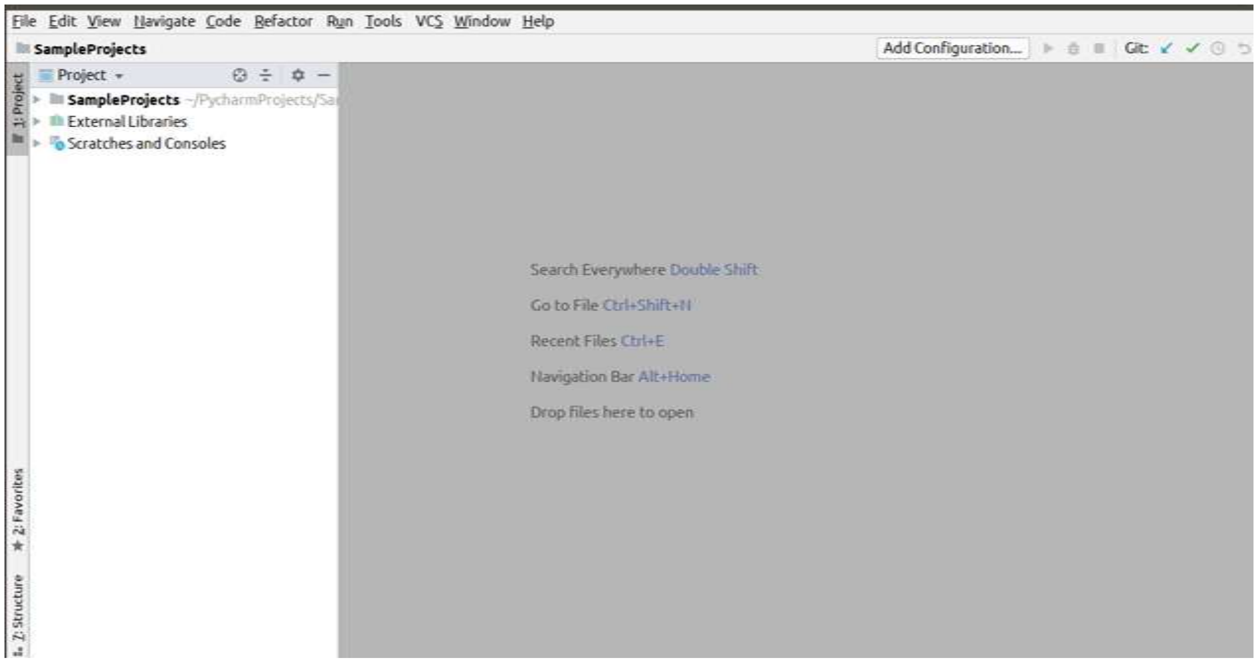
● Existing interpreter

Interpreter: <No interpreter>

Python 3.6 /usr/bin/python3.6

Click The *Create* Button

Here’s how the state of the IDE should look:



Create a Fresh Python File

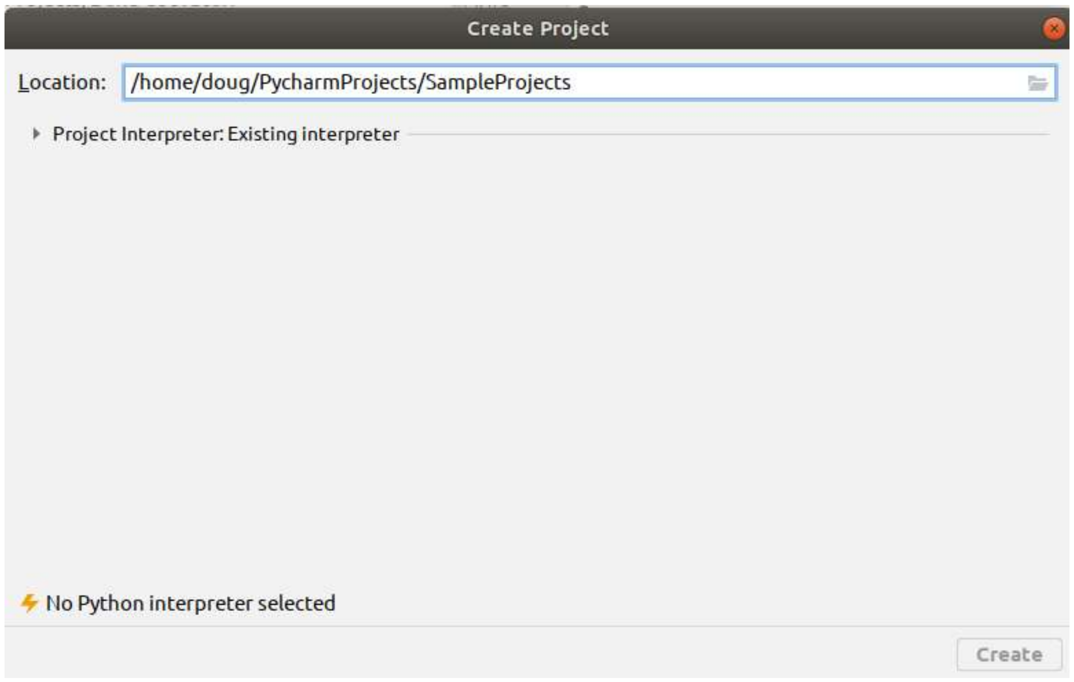
Create a new project by doing the following: *File* → *New Project*

What’s a Project in PyCharm

A project is an organizational unit in PyCharm. Name the project *SampleProjects*.

How a Project Looks Thus Far

Here's a screenshot of what the setup should look like thus far:



Location: /home/doug/PycharmProjects/SampleProjects

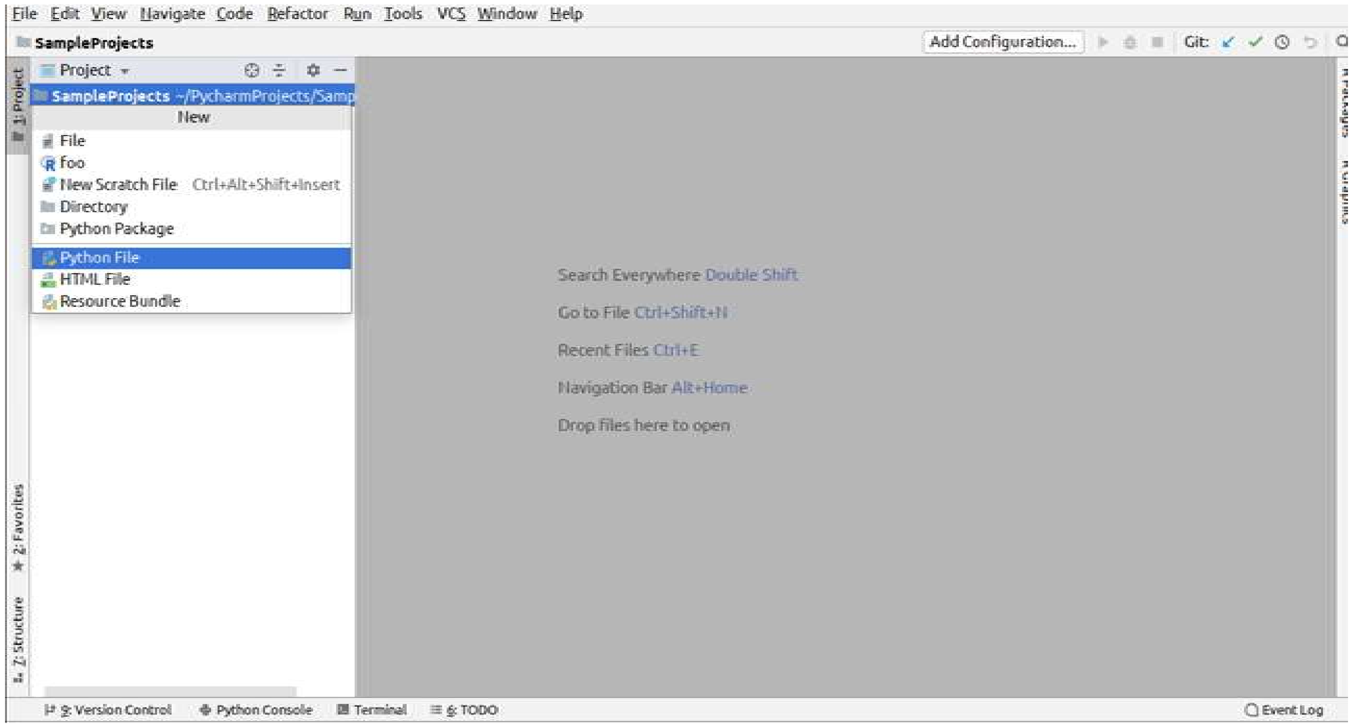
Project Interpreter: Existing interpreter

No Python interpreter selected

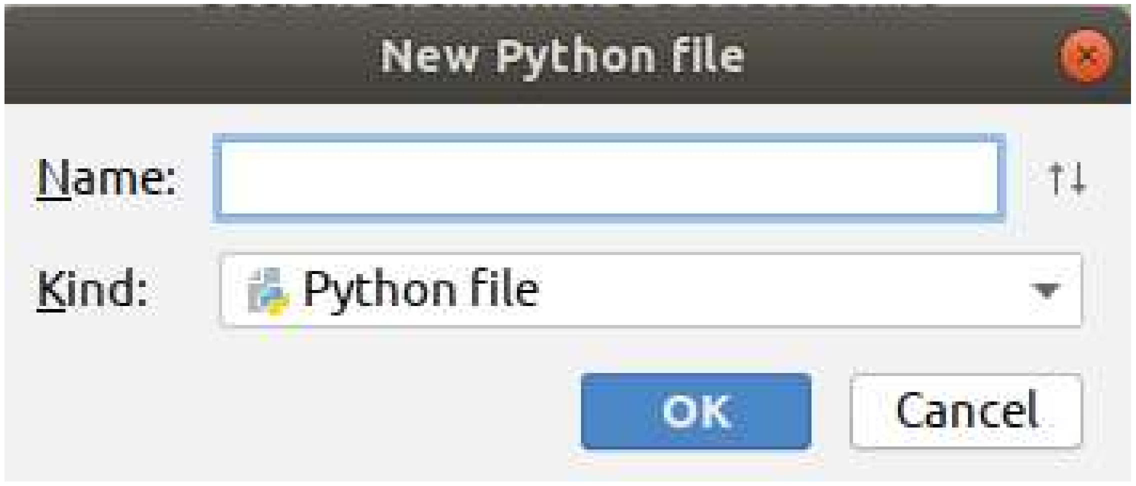
Create a Fresh Python File

At the main menu on PyCharm, or the portion where you see the various menus such as File, Edit, and View, do the following: **File → New File → python file**

Here’s How Pycharm Should Look

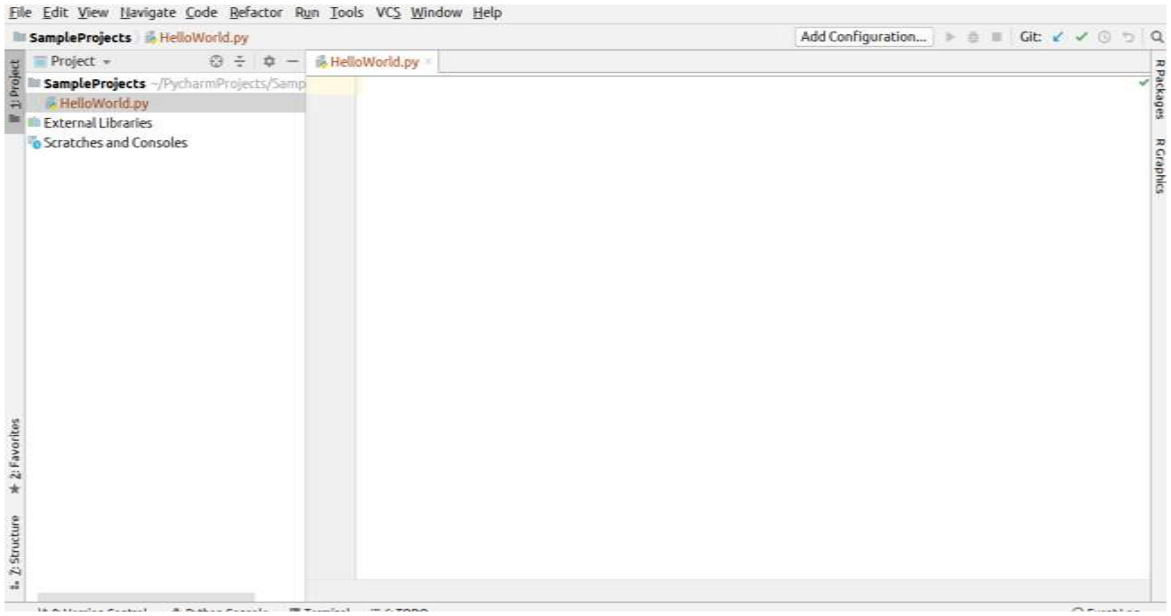


A Text Box Should Appear Which Looks Like the Following



Do the following...

Enter in the name *HelloWorld* and select the OK button. Once done here’s how your project setup should look:



What You’re Looking At

The blank white space is known as the **editor.** That’s where you’ll spend most of your time hacking away. The left hand side is known as the **project manager**, that’s where you can see the organization of files in a project.

Add Code and run the file

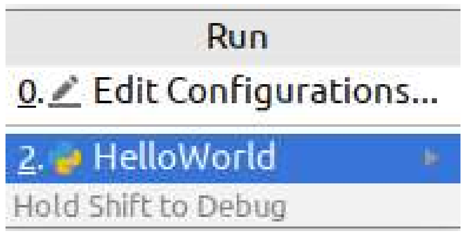
Copy the following code into the editor:

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

How to run the file

To run the file click the following on the main menu: run → run

A dialog box should appear which looks like the following:



Select the *HelloWorld* program. Once the program has executed you should see the text: *Hello World!!!*

If all went well then...

Congrats, you’ve ran your first python program.

If not, then raise your hand and the TA will provide you some one-on-one help.

Free Resources for Learning PyCharm

- Quick Start Guide: `https://www.jetbrains.com/help/pycharm/quick-start-guide.html`
- PyCharm blog: `https://blog.jetbrains.com/pycharm`
- Learn keyboard shortcuts for editing, navigating, refactoring, and debugging: `https://www.jetbrains.com/help/pycharm/mastering-keyboard-shortcuts.html`

Now That Your Environment is Setup, Let’s get to Some Interesting Stuff

Variables in Python

A variable in python is similar to a variable in mathematics. It’s something that has a changeable state.

Examples of Variables in Python

```
a = 10

b = 1.598

c = .1987

d = 100.579
```

Printing Output

```
print(a)

print(b)

print(c)

print(d)
```

Swapping Variables

```
x = 5

y = 10

z = 30

x, y, z = z, x, y

print(x)

print(y)

print(z)
```

Here’s the Output

```
30

5

10
```

Variable Naming Tips Cheat Sheet

- Variable names can have letters, numbers, and underscores.
- Can't use a reserved word like print.
- Be as descriptive as possible with your variable names. This reduces ambiguity and helps make your code more maintainable when other developers follow in after you.
- Python IS case sensitive so apple is not the same as Apple.
- Put constants or variables that value is fixed in all CAPS. I.e., `DAYS_OF_WEEK = 7`

Refer to PEP 8 For More Details

PEP is short for python enhancement proposals. Learn more about it here: **`https://www.python.org/dev/peps/pep-0008`**

Python Math Operators

Let’s look at some of the math operators available in python: +, -, *, **, /, //, %

Code Demo of Python Math Operators

```
print(10 + 10)

print(50 - 10)

print(10 * 10)

print(20 ** 2)

print(9 / 5)

print(8 // 3)

print(11 % 5)

print(1e10)
```

Here’s the Output

```
20

40

100

400

1.8

2

1

10000000000.0
```

How to Use Python as a Souped Calculator

Use the built in *math* module. You can use it by using the import statement.

Math Module Example

```
import math

print(math.log(1000000, 2))

print(math.sqrt(9))

print(math.cos(100) + math.sin(90) + math.tan(90))

print(math.pi**2 * math.e)

...

19.931568569324174

3.0

-0.23888487632000044

26.828366297560617
```

Strings in Python and Beyond

If you have ever sent an SMS text, used Facebook chat, or sent an email then you have used strings. A **string** is a sequence of characters wrapped in quotes; in python it could be single, double, or triple quotes. The single or double quotes can be used interchangeably. The triple quotes are typically used as doctrings, or comments inside methods, functions, or classes.

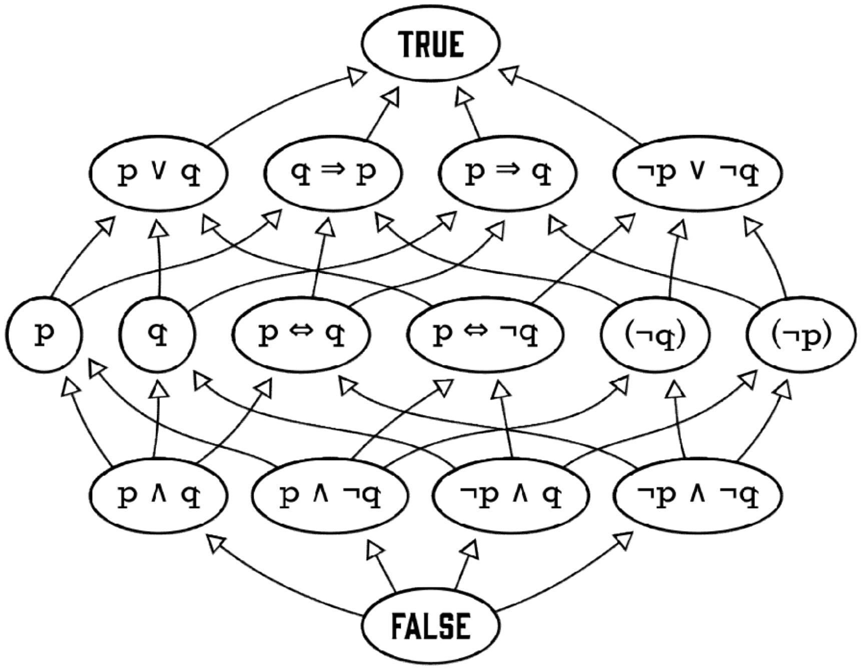
String Examples Python Part I

```
city = 'Los Angeles'

# indexing: python is a zero based indexed language

print(city[0]) # L

print(city[3]) # empty space is a string!

print(city[4]) # capital A

print(city[-1]) # negative indices are permitted

# len() function: gets the length of the string

print(len(city)) # 11

print(city[len(city)-1]) # s
```

String Examples in Python Part II

```
# concatenation: the combining of multiple strings

print('john ' + 'doe ' + 'public') # john doe public

# slicing: retrieves ranges of a string

print(city[0:3]) # Los

print(city[4:11]) # Angeles

print(city[::]) # Los Angeles

print(city[::2]) # LsAgls

print(city[::-1]) # selegnA soL
```

Boolean Algebra



Boolean Algebra is Prevalent in Many Languages

Learning Boolean algebra for python means that you can apply that set of logic to a wide array of languages like Java, C++, Haskell, Erlang, or R.

How Does It Work?

In python what we need to worry about are the values of `True` or `False`, which can also be denoted by 1 or 0 respectively. The main operations that we’ll discuss are “and” (conjunction), “or” (disjunction), as well as “negation.” There’s also the lesser used “xor” operator.

How the Truth Table Looks

Below is a sample of how the truth table looks:

x	y	x and y		x or y	not x
0	0	0	0	0	
1	0	0	1	1	
0	1	0	1	0	
1	1	1	1		

Boolean Algebra in Python Example # 1

```
is_the_sky_blue = True

do_cats_bark = False


print(is_the_sky_blue) # True

print(do_cats_bark) # False
```

Boolean Algebra in Python Example # 2

```
print(True and True) # True

print(True and False) # False

print(False and False) # False

print(False and True) # False
```

Boolean Algebra in Python Example # 3

```
print(True or True) # True

print(True or False) # True

print(False or False) # False

print(False or True) # True
```

Boolean Algebra in Python Example # 4

```
print(True ^ True) # False

print(True ^ False) # True

print(False ^ True) # True

print(False ^ False) # False
```

How to Next Use Boolean Algebra? Control Flow

Once you understand Boolean algebra you can apply that newfound knowledge to control flow, or the order in which statements are executed. Python uses the `if`, `else`, and `elif` statements for this.

`if/else` statement

```
x, y, z = 5, 10, 15

if x < y and z > y:

    print(x)

else:

    print(y)
```

elif statement

```
from random import randint

# picks a random number in range 1...100

grade = randint(1, 100)

if grade >= 90 <= 100:

    print('A')

elif grade >= 80 <= 89:

    print('B')

elif grade >= 70 <= 79:

    print('C')

elif grade >= 60 <= 69:

    print('D')

else:

    print('F!')
```

Ternary Statement

This is a special type of operator that evaluates something based on a condition being `True` or `False`.

```
mood = True

state = 'nice' if mood else 'not so nice'

print('state = {}'.format(state))
```

It prints `nice` because `if mood` evaluates to `True`.

Comments in Programming

At this point you may have saw the hash symbol (#) followed by text. This is known as a *comment* in python and this portion of the code is ignored by the interpreter. However, it’s still very useful to include in your programs as it helps other programmers that may be messing around in your code to understand the logic.

Examples of Comments

```
# This is a comment

# This is a comment and will be ignored by the interpreter

# I think you get the memo!
```

Data structures in python 3.7.3

There are four built-in data structures in python which are lists, tuples, dictionaries, and sets. Here’s a quick rundown of each one...

Lists

Mutable collections of objects.

List Demo 1

```
evens = [0, 2, 4, 6, 8, 10]

# reverses the list

>>> evens.reverse()

[10, 8, 6, 4, 2, 0]

# adds an object to the list

>>> evens.append(100)

[10, 8, 6, 4, 2, 0, 100]

# merges another list with the list

>>> evens.extend([1, 3, 5, 7, 9])

[10, 8, 6, 4, 2, 0, 100, 1, 3, 5, 7, 9]

# pops an item from the list

>>> evens.pop()
```

Tuples

These are an **immutable sequence**. Unlike lists once you create a tuple they cannot be modified, and trying to do so will cause an error.

Tuple Examples

```
>>> a = (5, 10)

>>> a

(5, 10)

>>> a[0] = 10

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

TypeError: 'tuple' object does not support item assignment

>>> a = (10, 20)

>>> a

(10, 20)
```

Tuples Are Immutable Therefore...

They don’t support item re-assignment. Here’s some more code snippets for better understanding:

```
>>> x = (5, 10)

>>> y = (5, 10)

>>> x == y

True

>>> x is y

False

>>> id(x), id(y)

(54484704, 54530096)
```

Dictionaries

These are key/value pairs or associative arrays in other languages. The concept is very similar to the dictionary reference source. You’re given a set of words which have corresponding definitions.

Dictionary Example in Python

```
vowels = {'a': 0, 'e': 0, 'i': 0, 'o': 0, 'u': 0}

>>> vowels.items()

dict_items([('a', 0), ('e', 0), ('i', 0), ('o', 0), ('u', 0)])

>>> vowels.get('a')

0

>>> vowels.values()

dict_values([0, 0, 0, 0, 0])

>>> vowels.keys()

dict_keys(['a', 'e', 'i', 'o', 'u'])

>>> vowels.pop('e')

0

>>> vowels

{'a': 0, 'i': 0, 'o': 0, 'u': 0}
```

Sets in Python

The set data structure stores unique items. Here’s a demo of a set in python:

A demo of sets in python

```
>>> letters = {'a', 'a', 'a', 'b', 'b', 'b'}

>>> letters.intersection({'b', 'c'})

{'b'}

>>> letters.pop()

'a'

>>> letters.add('c')

>>> letters.union({'a', 'e', 'i', 'o'})

{'i', 'c', 'e', 'b', 'o', 'a'}

>>> sorted(letters)

['b', 'c']
```

Iteration in Python

Iteration is the process in which computers do repetitive tasks. Humans despise repetition while computers are amazing at it. Humans can do repetitive tasks like summing all of the numbers from 1-100 manually (assuming no mathematical formulas are used), but these tasks are tedious and error prone. Computers can do number crunching like this in very quick times, like in a couple of freaking nanoseconds. There’s two ways to do iteration in python which is `while` or `for` loops.

While loop

```
# sets while loop starting at

i = 0

# condition

while i < 10:

    print('i = {}'.format(i, end=' '))

    i += 1 # increment i
```

The Output

```
0 ... 9
```

Another Example of a While Loop

```
# Sum numbers from 1...1000 in nanoseconds

i, sum = 0, 0

while i < 1000:

    i += 1

    sum += i


>>> print('The summation of 1...1000 = {}'.format(sum))

…

The summation of 1...1000 = 500500
```

`for` loop

This is another way to iterate in python. It can be used with the `range` function to iterate over a sequence of numbers or it can be used standalone to iterate over data structures like lists or sets.

`range()` example

```
for x in range(10):
    print(x, end=' ')
…
```

The above prints the numbers 0 … 9 on the same line separated by a space.

The `for` loop is also good for...

Iterating over data structures.

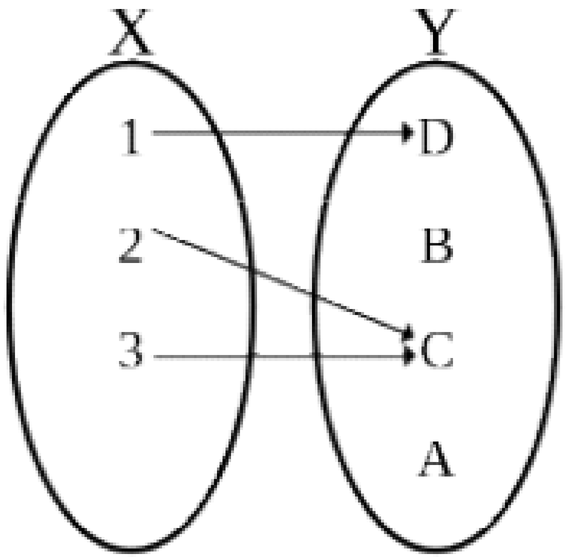
How to Iterate Over a List

```
>>> odds = [1, 3, 5, 7, 9]
>>> for x in odds:
...     print(x)
...
1
3
5
7
9
```

Fibonacci Numbers

```
x, y = 0, 1

for z in range(10):

    next = x + y

    x, y = y, next

print('12th fib number = {}'.format(next))

>>> print('12th fib number = {}'.format(next))

...

12th fib number = 89
```

A function in mathematics



How to create functions in python?

Use the `def` keyword.

Function Example in Python

```
def scale_number(num, amount):
        return num * amount

>>> print(scale_number(10, 5))

...

50
```

Keyword Arguments

```
def area_triangle(height=11, width=7.5):
        return 1/2 * (height * width)


>>> print(area_triangle())

...

41.25

>>> print(area_triangle(height=20, width=100))

...

1000.0
```

Accepting an arbitrary number of input

```
def multiply(*args, y=1):

for x in range(len(args)):

    y *= args[x]

return y



>>> print('multiply=', multiply(1, 2, 3, 4))

...

multiply= 24
```

Reading in an arbitrary number of keyword arguments

```
def key_value(**kwargs):

for key, value in kwargs.items():

    print('{} {}'.format(key, value))



>>> key_value(a=5, b=10, c=15)

a 5

b 10

c 15
```

Classes and Objects

Object oriented programming is a style of programming that involves the heavy use of classes and objects. Classes are typically described as blueprints, while objects are described as the templates that are created from the classes.

Let’s Model a Point

A **point** in mathematics refers to an element of some set called a space. Let’s model a point in python by using a class.

```python
class Point:
    """Simple class in python. This is an example
    of a docstring, or a string that's used like a
    comment to document a segment of code."""
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def get_x(self):
        return self.x

    def get_y(self):
        return self.y
```

```
def set_x(self, new_x):

    self.x = new_x


def set_y(self, new_y):

    self.y = new_y


def get_point(self):

    return self.x, self.y
```

How to Use a Class in Python?

Create an instance and then call it’s methods. Below is a simple example of how to create an instance of a class:

```
class Apples:

    pass



green = Apples()
```

Methods vs Functions

A method is binded to an object while a function is not:

```
def do_homework():

    return

...

class Schedule:

    def __init__(self):

        pass

    def do_homework(self):

        return
```

What do `__init__` do?

This is short for *initializer*, and it initializes the state of the class. Like with a method it can contain parameters which must match when an instance of the class is created.

The `self` convention

When you create a class in python all of the methods must have this convention. The `self` parameter is the first argument of every method in python and it’s used to pass the object.

Now, Back to the `Point` Class

```
p = Point(5, 10)

print()

print(p.get_point())

p.set_x(100)

p.set_y(200)

print(p.get_point())

...

(5, 10)

(100, 200)
```

Exception Handling

Exceptions are special ways in handling errors and unintended conditions in our program. In other words, it makes your program more robust and avoids crashes in programs.

A simple `try/except` statement

```
def divide(num, den):

    try:

        x = num / den

        print('{} / {} = {}'.format(num, den, num / den))

    except ZeroDivisionError:

        print("can't divide by zero.")

>>> divide(10, 5)

>>> divide(0, 10)

>>> divide(10, 0)

10 / 5 = 2.0

0 / 10 = 0.0

can't divide by zero.
```

The statement that wants to be executed is located within the `try` block. If an error occurs within the try block then the `except` block is executed. `ZeroDivisionError` is one of the many builtin exceptions in python3.

Another Example of a `try/except` statement

```
def import_test():

    try:

        import math

        import operating

        import sys

        print(math.pi)

        print(sys.version_info)

    except ImportError:

        print("Couldn't import something")

>>> import_test()

...

Couldn't import something
```

Explanation

The reason for this is because `operating` is not a built-in module in python and therefore an error was triggered while in the `try` block. You can also use the `raise` statement to force an exception to occur.

The `raise` statement

```
try:

    a = input('Enter an integer ')

    raise Exception("Something strange happened")

except ValueError:

    print("An exception happened.")



>>> Enter an integer 10

Traceback (most recent call last):

File "<stdin>", line 3, in <module>

Exception: Something strange happened
```

`try/except/finally` statement

```
def divide(a, b):

    try:

        result = a / b

    except ZeroDivisionError:

        print("Can't divide by 0")

    else:

        print(result)

    finally:

    print('This is in the finally statement')

>>> divide(10, 2)

...

5.0

This is in the finally statement
```

Image Credits

Python Interpreter: https://learning-python.com/class/Workbook/unit02.htm

Mosaic IDE: http://www.mosaic-industries.com/Products/Software/IDE.html

Boolean Algebra: https://www.investopedia.com/thmb/tYahC_el76R8oETvnxgfH9uVdZo=/1000x750/smart/filters:no_upscale()/Free-boolean-algebra-hasse-diagram-a30ae4a9499547d186f0343e31eda288.png

Functions: https://upload.wikimedia.org/wikipedia/commons/thumb/8/83/Injection_keine_Injektion_2a.svg/220px-Injection_keine_Injektion_2a.svg.png