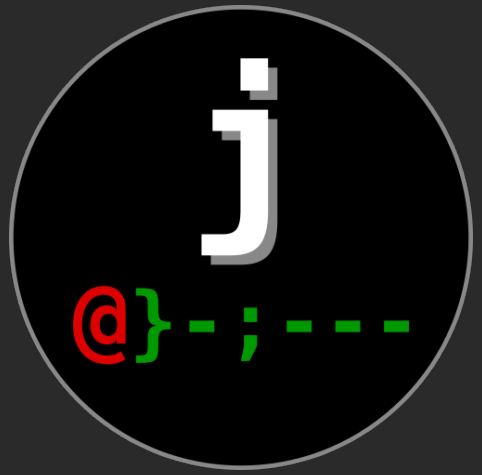
Foundational Python

Learn How to Code in an Hour



junktext.com

Lorem Ipsum

Presented by:
William Paul Liggett

Date and Place:
July 21st, 2018
The Circle of HOPE
New York City

$ whoami

Name: William Paul Liggett

Alias: junktext

Speaks: Python, Java, PHP, & HTML5/CSS3/JavaScript

Past: TA for Python, PHP/JavaScript Coder, Technical Proj. Manager, & Sys. Admin

.edu: M.S. in Information Sys. Tech (GWU)
Bachelor’s in Integrated IT (GWU)

What is Python?

* High-level computer language.
* Simple to learn, but super-powerful!
* Cross-platform:
  > Linux, Windows, macOS, others.
* Similar to Java, but with less rules and syntax.
* Python is interpreted and bytecode-compiled:
  > Source Code (.py) → Compiled CPython bytecode (.pyc)
  [NOT: Machine Language]

Who uses Python?

* TIOBE Index Ranking: #4 [1]
* Django: Mozilla.org, Instagram, & Pinterest [2]
* Plone: FBI, NASA, Harvard, PennState [3]
* Custom: Google [4], Raspberry Pi [5], Red Hat [6]
* *And lots of other cool/sexy people…*

[See: References]

References

[1] https://www.tiobe.com/tiobe-index, July 2018

[2] https://www.djangoproject.com/start/overview, July 2018

[3] https://plone.com/about/they-use-plone, July 2018

[4] https://github.com/google?language=python, July 2018

[5] https://www.raspberrypi.org/documentation/usage
/python/README.md, July 2018

[6] https://redhat.jobs/jobs/?q=python, July 2018

Programming Basics

Data vs. Information

? ?

Programming Basics

Data	vs.	Information
Input		Output

Writing Output (aka Info)

* **Output:** Something that comes *out* of the computer.
  - Simple: General print messages.
  - Advanced: Report based on 5 GB of data (input).

* **In Python:**

```
print("Hello, World!")
```

Outputs: Hello, World!

Variables

* **Variable:** Basically a cup to hold data.

  Allows your program to remember something. Holds one piece of data, which can change or vary later. Such as: x = 1 ... x = 5

* **In Python:**

```
cool_person = "William"  [Text needs "quotes".]
coffees_drank = 3        [Don't quote numbers.]
```

Python Variable Types

* **Integer:** Whole numbers.
  `a = 2` `b = 0` `c = -3`
* **Float:** Numbers with a *floating*-point decimal.
  `d = 1.5` `e = 0.45` `f = -94.2145`
* **String:** A *string* of characters. Text.
  `some_text = "ABCdef123 !@#$ YoYo :-)"`
* **Boolean:** True or False. [Note: Capital letters!]
  `amazing_dancer = True` `weakling = False`

Converting Variable Types

* Integer to Float: `float(2)` Output: 2.0
* Float to Integer: `int(2.0)` Output: 2
* Integer to String: `str(2)` Output: "2"
* Float to String: `str(2.0)` Output: "2.0"
* String to Integer: `int("2")` Output: 2
* String to Integer: `int("2.0")` Output: ERROR!
* String to Integer: `int("Bob")` Output: ERROR!

Getting User Input

* Using the input() function:

```
first_name = input("What is your first name? ")
```

* Then, if the user types: Bob

* We can then print a simple output message:

```
print("Hello,", first_name)
```

Outputs: Hello, Bob

Mathematical Operators

Example Variable: a = 5

* Add:	+	b = a + 2	b is: 7
* Subtract:	-	b = a - 2	b is: 3
* Multiply:	*	b = a * 2	b is: 10
* Divide:	/	b = a / 2	b is: 2.5
* Int Div:	//	b = a // 2	b is: 2
* Remainder:	%	b = a % 2	b is: 1
* Exponent:	**	b = a ** 2	b is: 25

Augmented Operators

```
                Example Variable:   a = 5
* Add & Assign:          +=        a += 1      a is: 6
* Subtract & Assign:     -=        a -= 1      a is: 4
* Divide & Assign:       /=        a /= 2      a is: 2.5
* Multiply & Assign:     *=        a *= 2      a is: 10
  > No ++ (increment) or -- (decrement) operators!
  > The += and the *= can also be used on strings:
text = "Blah"         text += "Yo"          text: "BlahYo"
```

Relational Operators

* Greater Than: `1 > 1` (False)
* Greater or Equal: `1 >= 1` (True)
* Less Than: `2 < 4` (True)
* Less or Equal: `0 <= 5` (True)
* Equal To: `3 == 3` (True)
* Not Equal To: `3 != 3` (False)

Boolean Operators

* and
* or
* not
* not()
* xor *[Not in Python!]*

Truth Table

```
True and True   =  True
True and False  =  False
True or True    =  True
True or False   =  True
not True        =  False
not(True)       =  False
```

Using Extra Python Modules

* “Batteries included.”
  > You can do a quick `import` statement to pull in a vast number of extra Python modules when needed.

```
import math
some_num = math.pi + 2/3 - 4.3
print(some_num)   Outputs: -0.49174067974354 02
```

Commenting Your Code

Seriously... comment your damn code.

```
# This is a single-line comment.

'''
This is a multi-line comment.
And here is some more text.
'''
```

```
'''
Builds the right-side that does not include this animal.
The following looks confusing but is simply:
  new_world_grid[row][string slice up to the edge of the World]
We do 'location+animal_width' to skip over what the i2 loop built above.
'''
build_line += new_world_grid[i+y_location][location+animal_width:World.WIDTH]
```

Algorithm vs. Program

* **Algorithm:** Step-by-step instructions (Pseudocode)

  1. Ask the person their name.
  2. Say hello to that person.

* **Program:** Step-by-step instructions (Source Code)

```
first_name = input("What is your first name? ")
print("Hello,", first_name)
```

Three Main Control Structures

Sequence

```
Step #1
Step #2
Step #3
```

Selection

```
if x > y:
    Step #A1
    Step #A2

else:
    Step #B1
```

Repetition

```
while x <= 10:
    Step #1
    Step #2
    x = x + 1
```

Sequence Example
Area of a Circle

```
# Used for Pi
import math

print("Area of a Circle")
print("-----------------\n")

# We need to use float() as input() is a string.
radius = float(input("Radius: "))

# Area = Pi(r^2)
area = math.pi * (radius ** 2)

print("Area =", area, "units squared")
```

Selection Example

Conditional: if, elif, else

```
first_name = input("What is your first name? ")

if first_name == "Bob":
    print("Hello and welcome,", first_name)

elif (first_name == "Joe") or (first_name == "Tom"):
    print("I hate you,", first_name)

else:
    print("I don't know you. Go away.")
```

Repetition Examples

while loop: Stops when a set condition is met.

```
x = ""

while x != "4":
  x = input("2 + 2 = ")

  if x != "4":
    print("No, try again!")

print("Correct!")
```

for loop: Stops after a known amount of times.

```
# Outputs: 0, 1, 2, 3
for i in range(0, 4):
  print(i)
```

```
# 10 character string
sentence = "Hi, there!"

for i in sentence:
  print("Runs 10 times.")
```

Functions

```
* Function: A reusable block of code.

def add_sales_tax(subtotal):
    # Adds 5% Sales Tax
    return subtotal + (subtotal * 0.05)

purchase_subtotal = 55.78
purchase_total = add_sales_tax(purchase_subtotal)

print("Subtotal:      $", purchase_subtotal)
print("Total (w/Tax): $", purchase_total)
```

Object-Oriented Programming

*** **PURPOSE:** Makes code represent real-world objects (people or cars) or abstract entities (e.g., checking accounts).

* **Class:** The blueprint to create objects.
* **Attributes:** Variables associated with an object.
* **Methods:** Functions that modifies an object.
* **Object:** An instance of a class. (student_1)
  > An object contains both attributes (data) and methods (internal functions).

OOP: Class Structure

```
class ClassName(object):
    # Constructor
    def __init__(self, param_1, param_2):
        self.attrib_1 = param_1
        self.attrib_2 = param_2

    def setter_method(self, param_1):
        self.attrib_1 += param_1

    def getter_method(self):
        return self.attrib_1
```

OOP: Class Example

```
class CheckingAccount(object):
    # Constructor
    def __init__(self, acct_num, balance):
        self.acct_num = acct_num
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount

    def get_balance(self):
        return self.balance
```

OOP: Objects

```
acct_1234 = CheckingAccount("1234", 20.00)
acct_9876 = CheckingAccount("9876", 0.00)

acct_1234.deposit(35.78)
acct_9876.deposit(0.42)

print("Account: 1234, Bal: $", acct_1234.get_balance())
print("Account: 9876, Bal: $", acct_9876.get_balance())
```

```
Account: 1234, Bal: $ 55.78
Account: 9876, Bal: $ 0.42
```

Homework!

Email me your solution (.py file)

Assignment: Create both a rectangle and a circle class. Define a `get_area()` method on each to output the calculations. Attributes needed for circles: `radius`, but for rectangles: `length * width`.

Bonus Points: Make a text-based menu to let the user choose which shape they want. And, you can use a loop to re-run the program until they type some sort of an exit signal (like `"Q"` for quit).

* **Email:** junktext@junktext.com No time limit :-)