PYTHON REVIEW
E1/E2 2021

VARIABLES + DATA TYPES

Python uses the same syntax for variable declarations and assignments.

```
# this is a comment
x = 12
y = 3.5
z = “hello!”

print(x + y)
print(z)
```

VARIABLES + DATA TYPES

Python variables do not have built-in types. If something is declared as an integer, for example, it can legally be reassigned a string value later.

```
# this is fine to Python
x = 12
x = “twelve”
x = True
```

VARIABLES + DATA TYPES

You are still allowed to cast your data values as needed.

```
# x <- ‘12’
x = str(12)
# y <- 12
y = int(12)
# z <- 12.0
z = float(12)
```

VARIABLES + DATA TYPES

Conveniently, Python lets you assign multiple variables at once.

```
# many to many
x, y, z = 1, 2, 3

# one to many
a = b = c = “hello”
```

COMMENTS

Inline comments in Python start with a '#' symbol.

Block comments are surrounded by three double quote marks.

```
# inline comment
x = 12 # another one

"""
This is how to write a
block comment in Python
"""
```

OUTPUT VARIABLES

Strings can be concatenated with a ‘+’ symbol.

To print multiple values at once, separate the arguments with commas.

```
a = “Hello“
b = “world!”
print(a + b)

ans = 1234.5
print(“answer is”, ans)
```

CONDITIONALS

Unlike many other languages, Python cares about whitespace.

Everything indented after the conditional is what will be executed.

```
x, y = 4, 10
if x > y:
    print(x, “is greater”)
elif x < y:
    print(y, “is greater”)
else:
    print(“equal values”)
```

LOOPS

While loops are done very standardly in Python, not much to say here.

```
i = 1
while i < 10:
    print(i)
    i += 1
```

LOOPS

For loops are used primarily for iterating over sets of data, but they can also be used like traditional counting loops.

```
for x in “hello!”:
    print(x)

nums = [2, 3, 4]
for x in nums:
    print(x ** 2)
```

FUNCTIONS

Python syntax is nothing if not consistent. Don’t forget that indents matter!

```
def hello():
    print(“Hello world!”)

def greet(name):
    print(“Hello”, name)
```

FUNCTIONS

Here’s an example function I wrote for PH1110’s first lab, might be useful for you.

```
import math

def u_area(x,dx,y,dy):
     area = x * y
     x_comp = (dx / abs(x)) ** 2
     y_comp = (dy / abs(y)) ** 2
     ans = area * math.sqrt(x_comp +
          y_comp)
     return ans

print(u_area(5.3, 0.1, 8.5, 0.1))
```

MODULES

Import them at the top of your source file like usual. For modules with longer names, you can give them an alias for easy reference later on.

```
# common modules
import math
import numpy as np
import matplotlib.pyplot
as plt
import scipy.stats as st
import pandas
```

ARRAYS

Vanilla Python lists are fine for most applications, but Numpy provides an array data structure with far more utility for scientific applications.

```
import numpy as np

x = np.array([1,2,3])
grid = np.array([1,0],[0,1])

# mean and standard deviation
print(np.mean(x))
print(np.std(x))
```

ARRAYS

Here’s an example for how to import data from a CSV file into a numpy array

```
import numpy as np

data = np.genfromtext(
    ‘data_file.csv’,
    delimiter=’,’)
```

ARRAYS

```
In [5]: import numpy as np

        data = np.genfromtxt('data.csv', delimiter=',')
        # deletes the 0th row
        data = np.delete(data, 0, 0)

        print(data)
```

```
[[5.3 8.5]
 [5.4 8.7]
 [5.3 8.5]
 [5.1 8.6]
 [5.2 8.9]
 [6.  8.4]]
```

ARRAYS

The axis argument in these functions specifies whether to analyze by row or column.

```
mean = np.mean(data, axis=0)
stdv = np.std(data, axis=0)

print("x = ", mean[0], "+/-", stdv[0])
print("y = ", mean[1], "+/-", stdv[1])
```

```
x =  5.383333333333333 +/- 0.2910708199428831
y =  8.6 +/- 0.16329931618554516
```

MORE RESOURCES

- W3schools - easy, readable reference for basic Python functionality
- Numpy Docs - reference manual for Numpy
- Lab GitHub - Example code for many of the labs, feel free to use and modify
- Talk to your classmates!
- Google it!

Thanks!

ANY QUESTIONS?

Email me:
anishimura@wpi.edu

Or message me on Slack :)