COMP1531

1.2 - Git, Python

Python

Interpreted V compiled

C program $\xrightarrow{\text{Compile}}$ Machine Code $\xrightarrow{\text{Run}}$ Output

Python program $\xrightarrow{\text{Compile \& Run}}$ Output

Python

python2

python3

PYTHON 2		PYTHON 3
← **Legacy**	>	**Future** →
It is still entrenched in the software at certain companies		It will take over Python 2 by 2020
Library	≠	**Library**
Many older libraries built for Python 2 are not forwards-compatible		Many of today's developers are creating libraries strictly for use with Python 3
0100 0001 **ASCII**	+	**Unicode** 0000 0000 0100 0001
Strings are stored as ASCII by default		Text strings are Unicode by default
5/2=2	≠	**5/2=2.5** =
It rounds your calculation down to the nearest whole number		The expression 5 / 2 will return the expected result
print "hello"	≠	**print ("hello")**
Python 2 print statement		The print statement has been replaced with a print () function

Python

Resources shown in this course will be from around the internet. Python has some of the best online-help of any language ever made.

Python

CLI (Command line interface)

- Can be run inline
- Can be run as cli entry
- Can be run via a file

```
hsmith@vx1:~$ python3
Python 3.7.3 (default, Apr  3 2019, 05:39:12)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello")
Hello
>>>
hsmith@vx1:~$ python3 -c 'print("Hello")'
Hello
hsmith@vx1:~$ 
```

Python - Basics

Basic code (basics1.py)

```python
name = "Giraffe"
age = 18
height = 2048.11 # mm

num1 = 3 ** 3
num2 = 27 // 3

print(name + ", " + str(age) + ', ' + str(height))
print(name, age, height, sep = ', ')
print(f"{name}, {age}, {height}")
print(type(name))
print(type(age))
print(type(height))
print(f"3 ** 3 == {num1}")
print(f"27 // 3 == {num2}")
```

- Garbage collection
- More info on data types

Python - Basics

Strings (basics2.py)

```python
1    sentence = "My"
2    sentence = sentence + " name is"
3    sentence += " Pikachu"
4
5    print(sentence)
6
7    print("Hi!!" * 10)
```

Python strings are **immutable**

Python - Basics

Control structures, argc/argv (basics3.py)

```python
import sys

argc = len(sys.argv)

empty = True
if argc > 0:
    empty = False

if not empty:
    if argc == 2:
        print("Nearly there")
    elif argc == 3:
        if sys.argv[1] == "H" and sys.argv[2] == "I":
            print("HI to you too")
        else:
            pass
else:
  print("Please enter two letters as command line")
```

.

Python - Basics

Lists, loops (basics4.py)

```python
names = [ "Hayden", "Rob", "Isaac" ]
names.append("Vivian")

for name in names:
    print(name)

print("===")

names += [ "Eve", "Mia" ]
for i in range(0, len(names)):
    print(names[i])
```

Python lists are very complicated arrays under the hood. You can read a lot here, here, and here.

Python - Basics

Tuples (basics5.py)

```python
1  x = 5
2  y = 6
3  point = (x, y)
4  print(point)
5
6  a, b = point # destructuring
7  print(f"{a}, {b}")
8
9  names = [ "Giraffe", "Llama", "Penguin" ]
10 for id, name in enumerate(names):
11   print(f"{id} {name}")
```

- lists are mutable, tuples are immutable

Git

Stage 1. Version control:

- clone
- status
- diff
- add
- commit
- push
- pull

Git

Stage 2. Teamwork:

- same as before
- log
- pull before you push
- merge (pull) requests
- merge conflicts