

ART WITH PYTHON



TURTLE

ANNOUNCEMENT

- Homework 2 will be posted today after TA William's tutorial.

LEARN

ITERATIONS AND RECURSIONS

PYTHON

JUST A LITTLE BIT MORE
CODING BASICS

Credit: lecture notes modeled after <http://www.openbookproject.net/thinkcs/python/english2e/index.html>

for Loop

<http://www.pythontutor.com/index.html>

Syntax

```
for iterating_var in sequence:  
    _____STATEMENTS
```

```
fruit = 'apple'  
for each_char in fruit:  
    print each_char
```

e - p a

Range function

```
A=range(10)
B=range(2,7)
C=range(0,10,2)
D=range(-10, -30, -5)
print A
print B
print C
print D
```

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

[2, 3, 4, 5, 6]

[0, 2, 4, 6, 8]

[-10, -15, -20, -25]

```
a = ['I', 'love', 'Python', 'programming']  
for i in range(len(a)):  
    print i, a[i]
```

01

1 love

2 Python

3 programming

while Statement

Syntax

```
while EXPRESSION:  
    _____STATEMENTS
```

```
def newyear_countdown(n):  
    while n > 0:  
        print n  
        n = n-1  
    print "Happy New Year!"  
newyear_countdown(10)
```


10

9

8

7

6

5

4

3

2

1

Happy New Year!

```
def num_digits(n):  
    count = 0  
    while n:  
        count = count + 1  
        n = n / 10  
    return count  
print num_digits(54320)  
print num_digits(int('012345'))  
print num_digits(int(23.45))  
print num_digits(23.45)
```

5

5

2

And an infinite loop

```
def print_powers(n):  
    i = 1  
    while i <= 6:  
        print n ** i, '\t',  
        i += 1  
    print  
print_powers(2)
```

2 4 8 16 32 64

Lists

List is an ordered set of values.
It can contain mixed types.

```
A = [1, 2, 3, 4]
```

```
print A
```

```
B = ["hello", "and", "good morning"]
```

```
print B
```

```
C = ["hello", 100, 'person', 2.5, [1, 2]]
```

```
print C
```


[1, 2, 3, 4]

['hello', 'and', 'good morning']

['hello', 100, 'person', 2.5, [1, 2]]

```
empty = []  
print empty  
print 'this is[' ,empty,']'
```

```
[]  
this is[[]]
```

```
empty = []  
print empty  
print 'this is[' ,empty,']'  
print type(empty)
```

```
[]  
this is[[]]  
<type 'list'>
```

```
numbers = [1, 2, 3, 4, 5, 6]  
print numbers[0]  
print numbers[5]  
print numbers[-1]  
print numbers[-2]
```

1
6
6
5

```
rainyday = ["today", "is", "a", "rainy", "day"]
```

```
i = 0
```

```
while i < len(rainyday):
```

```
    print rainyday[i]
```

```
    i += 1
```

```
#See the flow of the program
```


today
is
a
rainy
day

```
rainyday = ["today", "is", "a", "rainy", "day"]
```

```
print "today" in rainyday
```

```
print 'Today' in rainyday
```

```
print 'is' in rainyday
```

True
False
True

```
a = [1, 2, 3]
```

```
b = [10, 20, 30]
```

```
c = a + b
```

```
print c
```

```
d = a[1]*b
```

```
print d
```

```
e = a*4
```

```
print e
```

[1, 2, 3, 10, 20, 30]

[10, 20, 30, 10, 20, 30]

[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]

```
alphabet = ['a', 'b', 'c', 'd', 'e', 'f']  
print alphabet[1:3]  
print alphabet[:]  
print alphabet[0:1]  
print alphabet[0:8]
```

['b', 'c']

['a', 'b', 'c', 'd', 'e', 'f']

['a']

['a', 'b', 'c', 'd', 'e', 'f']

Lists are mutable.


```
pie = ["banana", "apple", "pear", "strawberry"]  
pie[0] = "peach"  
pie[-1] = "chocolate"  
print pie
```

['peach', 'apple', 'pear', 'chocolate']

```
if []:  
    print "empty"  
else:  
    print "full"
```

```
# [] acts like 0 here
```

full

Lists are mutable, strings are not.
Strings are immutable.

Tuple is a sequence of items of any type.
Tuple is immutable.

```
my_tup = (1, 2, 3, 4, 5)
print type(my_tup)
print my_tup[0]
my_tup[0] = 6 #Assignment is not supported
```

```
<type 'tuple'>
```

```
1
```

```
TypeError: 'tuple' object does not support item assignment
```


Recursion

```
def recursive_sum(nested_num_list):
    sum = 0
    for element in nested_num_list:
        if type(element) == type([]):
            sum = sum + recursive_sum(element)
        else:
            sum = sum + element
    return sum
```

```
print recursive_sum([1,2,3,4])
```

Tail Recursion

```
def newyear_countdown(n):  
    if n == 0:  
        print "Happy New Year!"  
    else:  
        print n  
        newyear_countdown(n-1)  
newyear_countdown(5)
```

5

4

3

2

1

Happy New Year!

#see the order of execution

More Recursion Examples

```
def factorial(n):  
    if n == 0:  
        return 1  
    else:  
        return n * factorial(n-1)  
print factorial(4)
```

Fibonacci number



Credit: Wikipedia


```
def fibonacci (n):  
    if n == 0 or n == 1:  
        return 1  
    else:  
        return fibonacci(n-1) + fibonacci(n-2)  
  
print fibonacci(3)
```

3

Python Turtle Review

<https://trinket.io/python>

```
import turtle
johnny = turtle.Turtle()
for i in range(0,4):
    johnny.forward(100)
    johnny.right(90)
```

<https://trinket.io/python>

```
import turtle
def draw_polygon(sides, length):
    johnny = turtle.Turtle()
    for i in range(0,sides):
        johnny.forward(length)
        johnny.right(360/sides)
```

```
draw_polygon(4,20)
```

```
draw_polygon(6,20)
```

```
import turtle
def draw_spiral(angle, length_start, length_increase, sides):
    for i in range(0,sides):
        johnny.forward(length_start+(i*length_increase))
        johnny.right(angle)

johnny = turtle.Turtle()
draw_spiral(30, 10, 2, 20)
```

```
import turtle
def draw_petals(length, number):
    for i in range(0, number):
        johnny.forward(length)
        johnny.right(180-(360/number)) # number divisible by 360
johnny = turtle.Turtle()
draw_petals(50,20)
```

Credit: <https://www.linuxvoice.com/issues/002/02drawing.pdf>

Recursion with Python Turtle

<https://trinket.io/python>


```
import turtle
```

```
myTurtle = turtle.Turtle()
```

```
myWin = turtle.Screen()
```

```
def drawSpiral(myTurtle, lineLen):
```

```
    if lineLen > 0:
```

```
        myTurtle.forward(lineLen)
```

```
        myTurtle.right(90)
```

```
        drawSpiral(myTurtle, lineLen-5)
```

```
drawSpiral(myTurtle, 100)
```

Credit:

<http://interactivepython.org/runestone/static/pythonds/index.html>

```
import turtle
```

```
def tree(branchLen,t):  
    if branchLen > 5:  
        t.forward(branchLen)  
        t.right(20)  
        tree(branchLen-15,t)  
        t.left(40)  
        tree(branchLen-15,t)  
        t.right(20)  
        t.backward(branchLen)
```

Credit:

<http://interactivepython.org/runestone/static/pythonds/index.html>

```
def main():  
    t = turtle.Turtle()  
    myWin = turtle.Screen()  
    t.left(90)  
    t.up()  
    t.backward(100)  
    t.down()  
    t.color("green")  
    tree(75,t)  
    myWin.exitonclick()
```

```
main()
```

Credit:

<http://interactivepython.org/runestone/static/pythonds/index.html>

```
from turtle import *

def drawSnowFlake(length, depth):
    if depth > 0:
        for i in range(6):
            forward(length)
            drawSnowFlake(length // 3, depth - 1)
            backward(length)
            left(60)

drawSnowFlake(60,2)
drawSnowFlake(60,3)
```

WILLIAM'S TUTORIAL
ON HIS PYTHON TURTLE

PLAY WITH PYTHON
LABS ON YOUR OWN!



THANKS!

Any questions?

You can find me at
beiwang@sci.utah.edu

<http://www.sci.utah.edu/~beiwang/teaching/cs1060.html>

CREDITS

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by [SlidesCarnival](#)
- Photographs by [Unsplash](#)