

Hunter - Curriculum Development Unit Plan

Introduction to Python Programming

Week 1

Print Statements
Operators
Randomize Numbers

Deliverable

Students will create a Number
Guessing Game

Day 1

Print Statements

Objectives:

Print Statements

Variables

Text Input

Concatenate Strings

Print Statements

Python print() function

The print() function in Python is used to print a specified message on the screen. The print command in Python prints strings or objects which are converted to a string while printing on a screen.

Print Statements

Python print() function

Practice typing out some statements in the editor part of the IDE, then hit “Run” at the top of the screen:

```
print("Hello World")
```

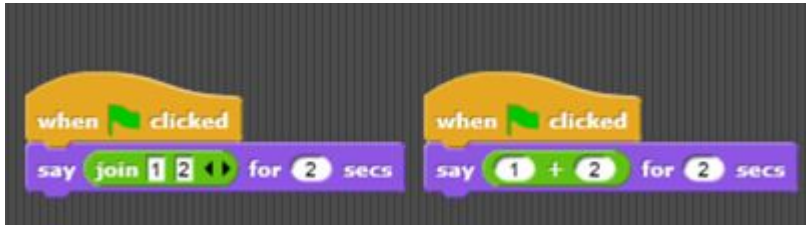
```
print("1")
```

```
print("this" + " " + "is" + " " + "a" + " " + "sentence" + ".")
```

Print Statements

Python compared to Snap

Print() in Python is similar to the "Say" block in Snap



Code



Image

Variables Practice

Variable: a name that refers to a value.

An assignment statement creates new variables and gives them values

```
>>> message = 'And now for something completely different'
```

```
>>> n = 17
```

```
>>> pi = 3.1415926535897932
```

Which are the variables, and which are the values?

Assignments work from right to left, so the item on the right is assigned to the item on the left.

Variables Practice

Type and run the following in your console

```
animal = "dogs"
```

```
print(animal + " are really cool.")
```

Type and run the following in your console

```
print(dogs + " are cool.")
```

Discuss

1. What happens?
2. Why?
3. How would you make the program print out "cats are really cool" instead?

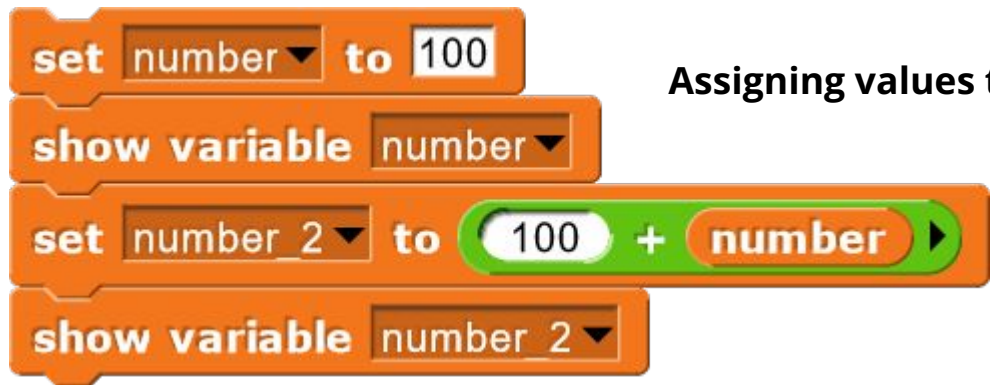
1. What was the output?
2. Why does this happen?

Variables Practice

Python compared to Snap

In Snap we name a variable and then set the value.

In Python we assign the value of the variable once it has been created.



The image shows four Scratch code blocks stacked vertically. The first block is 'set number to 100'. The second block is 'show variable number'. The third block is 'set number_2 to 100 + number'. The fourth block is 'show variable number_2'. To the right of the first two blocks is the text 'Assigning values to variables'.

Assigning values to variables

Getting Text Input

Type and run the following code:

```
a = input("What is your name? ")
```

```
# a = "cats and dogs"
```

```
# meow
```

```
print("Hello there, " + a)
```

Read through the code and discuss what you expect the printed results to be?

Discuss the actual printed result

Briefly describe what the # does

Briefly describe what input does

Getting Text Input

Discussion

What did input do?

What did the string between the parentheses do?

What would changing that string do? (it changes the prompt given to the user)

Input: user data given to the app.

Input lets a program take input from the console and use it in the code.

Mutability: you can change an existing value

Concatenate Strings

String concatenation means add strings together.

Use the + character to add a variable to another variable: Try these

```
x = "Python is "
```

To add a space between them, add a " "

```
y = "awesome"
```

```
a = "Hello"
```

```
z = x + y
```

```
b = "World"
```

```
print(z)
```

```
c = a + " " + b
```

```
print(c)
```

Concatenate Strings

String concatenation means add strings together.

Use the + character to add a variable to another variable: Try these

For numbers, the + character works as a mathematical operator:

```
x = 5
```

```
y = 10
```

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

If you try to combine a string and a number, Python will give you an error:

```
x = 5
```

```
y = "John"
```

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

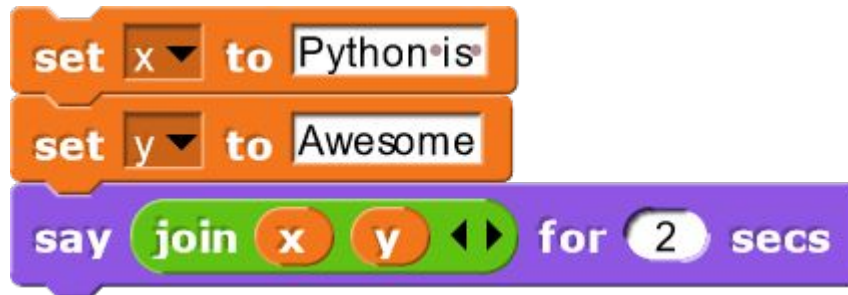
Concatenate Strings

Remember that in Snap when we wanted to combine variables and/or phrases we used the Join block.

The join block was used to concatenate strings and integers



We set variables in Snap and then joined them together



Day 2

Operators

Objectives:

Arithmetic Operators

Simple Conditional Statements

Arithmetic Operators

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication and division.

Arithmetic Operators

There are 7 arithmetic operators in Python

Operator	Description	Syntax
+	Addition: adds two operands	<code>x + y</code>
-	Subtraction: subtracts two operands	<code>x - y</code>
*	Multiplication: multiplies two operands	<code>x * y</code>
/	Division (float): divides the first operand by the second	<code>x / y</code>
//	Division (floor): divides the first operand by the second	<code>x // y</code>
%	Modulus: returns the remainder when first operand is divided by the second	<code>x % y</code>
**	Power : Returns first raised to power second	<code>x ** y</code>

Arithmetic Operators

Print the following operators to see how they work

```
print(5 + 3)
```

```
print(5 - 3)
```

```
print(5 * 3)
```

```
print(12 / 3)
```

```
print(15 // 2)
```

```
print(5 % 2)
```

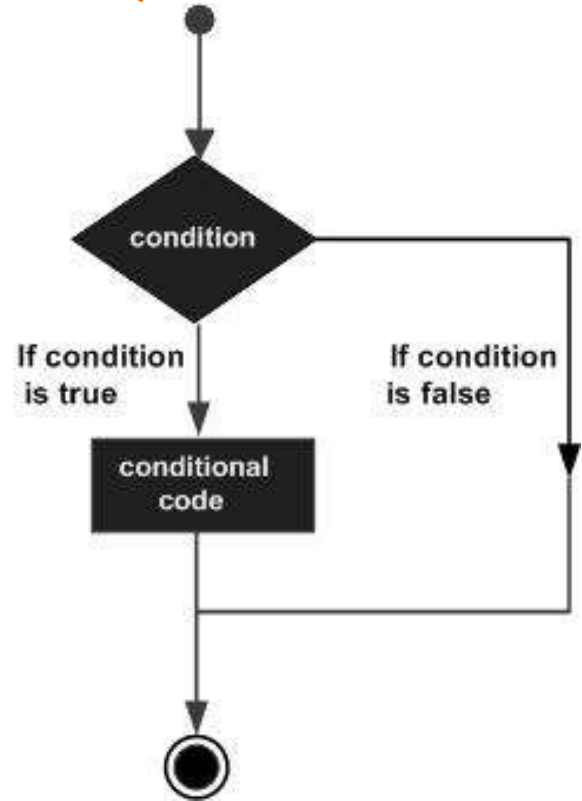
```
print(5 ** 2)
```

These operators are simple ones. If you have more than one operator make sure that you are following the order of operations.

Conditional Statements (Making Decisions)

Decision making is the anticipation of a given conditions that occurs running your program. It will specify actions to be taken taken based on the conditions.

We will evaluate multiple expressions which produce a TRUE or FALSE result for the outcome. We can determine which action to take and which statements to execute if outcome is TRUE If it is FALSE it will follow another path.



Conditional Statements

1) **if statements**

An if statement consists of a boolean expression followed by one or more statements.

2) **if...else statements**

An if statement can be followed by an optional else statement, which executes when the boolean expression is FALSE.

3) **nested if statements**

You can use one if or else if statement inside another if or else if statement(s).

Conditional Statements (if)

Run the following code to see how conditionals work

```
num = 21
```

```
if ( num == 21 ) : print "The student is 21 years old"
```

```
print "Good bye!"
```

Conditional Statements (if...else)

Run the following code to see how conditionals work

```
num = 3
```

```
if num >= 0:
```

```
    print("Positive or Zero")
```

```
else:
```

```
    print("Negative number")
```

Change the value of the variable and describe what happens

```
# num = -5
```

```
# num = 0
```

Conditional Statements (Nested Statements)

We can have a if...else statement inside another if...else statement. This is called nesting and is the same idea we used in Snap.

Since we can nest any number of these statements inside one another, they can get confusing. In Python, Indentation is the best way to figure out the level of nesting.

Let's write a program that tells you if a number is positive, negative, or is zero.

We will ask for the user to enter a number and then determine the value.

Conditional Statements (Nested Statements)

```
num = float(input("Enter a number: "))  
if num >= 0:  
    if num == 0:  
        print("Zero")  
    else:  
        print("Positive number")  
else:  
    print("Negative number")
```

Change the value of
the variable and describe
what happens

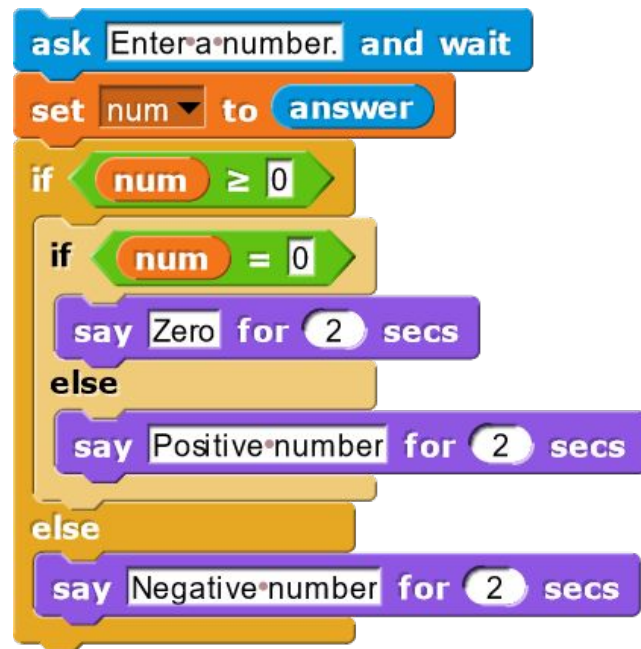
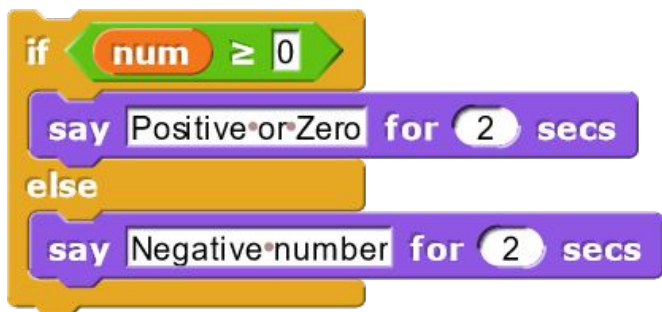
num = -5

num = 0

num = 5

Conditional Statements (Snap Reference)

Running the same code in Snap



Day 3

Random Numbers

Objectives:

Generating Random Numbers

Generate Random Numbers

The **randint()** function of the random module returns a random number between two given numbers. Your program can access randint() when you import the random module.

To get a random number between 1 and 10, pass 1 and 10 as the first and second **arguments**, respectively.

The following code will return a random integer between 1 and 10 each time you run the program (**including 1 and 10**).

Generate Random Numbers

```
import random  
  
random_number = random.randint(1, 10)  
  
print(random_number)
```

Run the code a few times to see what happens. Discuss with your neighbor.

Remember that in Snap we generated random numbers using the Operator block **"Pick Random"**



Input Max Value - Output Random Value

In Python you will want to get information from the end-user by asking questions and retrieving user inputs.

One way is to use the `input()` function.

```
username=input("What is your username?")
```

Whole numbers (numbers with no decimal place) are called integers. To use them as integers you will need to convert the user input into an integer using the `int()` function.

```
age=int(input("What is your age?"))
```

Input Max Value - Output Random Value

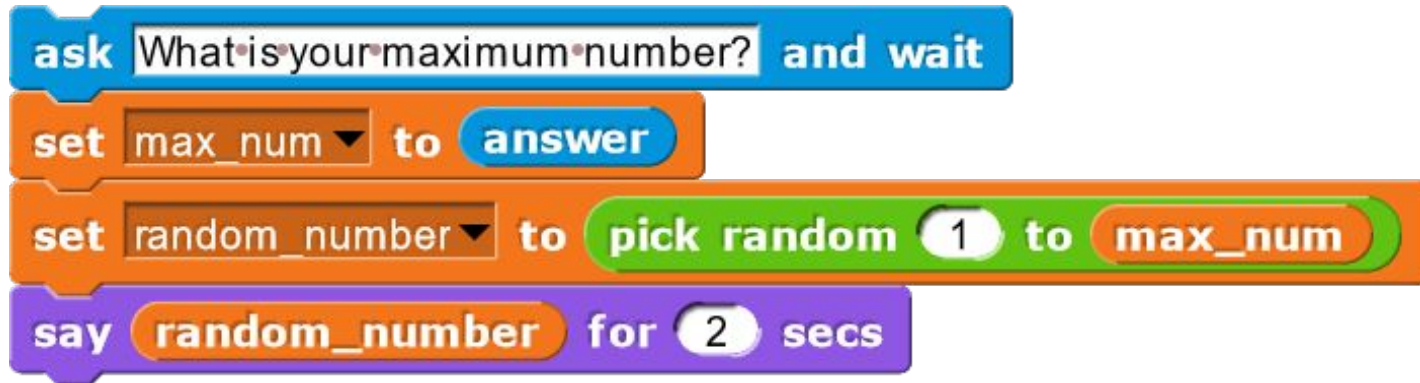
We will combine the ideas of the user giving a value and Python generating a random number. Let's ask the user for a maximum value and then generate a random number based on their input. Try the following code and explain what is happening.

```
import random  
  
max_value=int(input("What is your maximum number?"))  
  
random_number = random.randint(1, max_value)  
  
print(random_number)
```

Run the code
a few times and
change the
max_value.
Discuss with
your neighbor.

Input Max Value - Output Random Value

When we set random numbers in Snap we were also able to ask the user to input a value as the maximum value for the random integers.



Dice Simulator

```
import random
```

Import the random module

Set each die to generate a random integer between 1 and 6 - inclusive.

Print the value of each die

Set the dice total to a variable
print the dice total

Pull down to reveal code

Days 4-5

Lab: Number Guessing Game

Objectives:

Generate Random Numbers



Co-conspirators:

Usman Ahmed

Ed Hawkins

Kirk Martin

Wayne Tobias