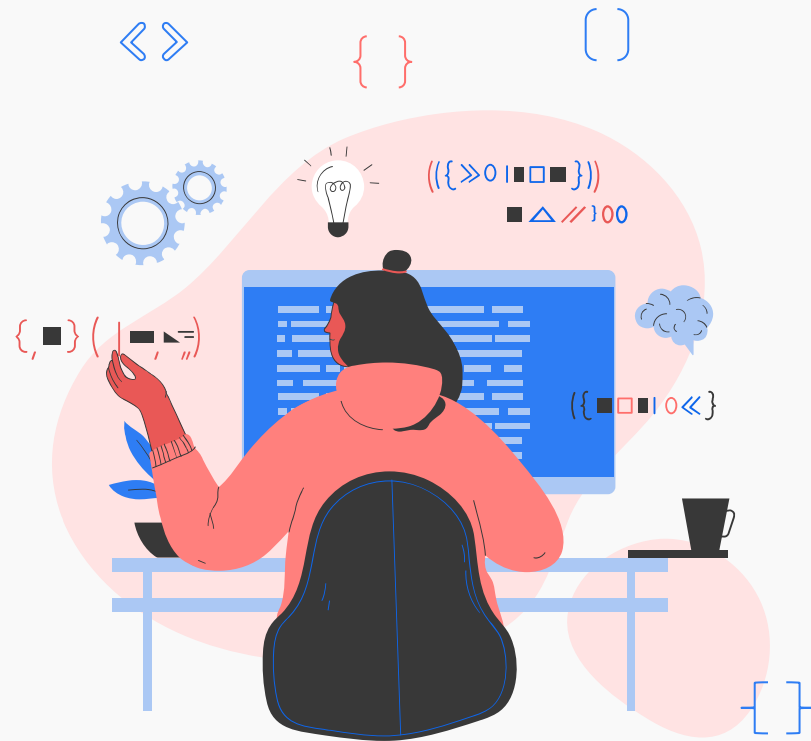


# Basics of Pythons

Intro to Coding on Python

Bernadette and Astrid





# Quick Summaries

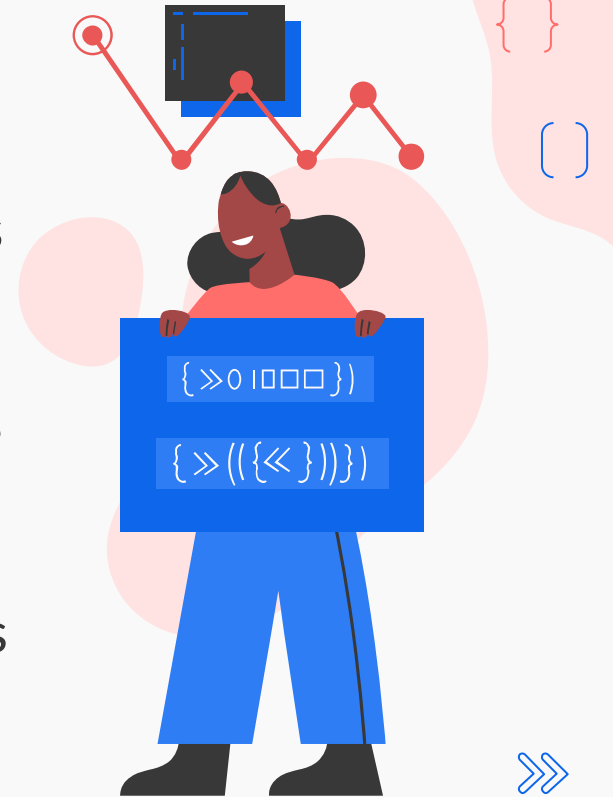
We will expand on these later but just to get these fresh in your mind

<b>Imports</b>	Statements that transfer code from other files to your own with pre-made functions
<b>Conditionals</b>	Allow programs to make decisions and display different paths for code to go
<b>Loops</b>	Makes it so a specific part of a program can be repeated a set amount of times
<b>Operators</b>	Help navigate code and are connectors of specific code segments
<b>Basic Format</b>	Code will not work unless colons, indentation, and spacing are right
<b>Functions</b>	Segments of code saved in a program to be used multiple times without rewriting the code again
<b>Lists</b>	Variables grouped together to hold information that can be later accessed



# About Python

Python is a high level programming language, meaning that it is designed to be easily understood by humans. Since computers process information using binary code(1001000 1100101 1111001 100000 1100010 1100001 1100100 1100100 1101001 1100101) people developed “languages” like Python or Java to allow for more complex and logical algorithms to be translated into binary. To sum it all up, Python is used to express these ideas in a way that both computers and humans can understand.





# Terms you need to know



## Program

A collection of program statements that performs a specific task when run by a computer. A program is often referred to as software.

## Parameters

The input variables of a procedure. A set of information included to help a process or function operate as intended.

## Variable

An abstraction inside the program that can hold a value. Each variable has associated data storage that represents one value at a time, but that value can be a list or other collection that in turn contains multiple values.

## Algorithm

A finite set of instructions that accomplish a specific task. An algorithm can be expressed in many kinds of notation, such as natural language, pseudocode, and flowcharts. Algorithms are essential to the way computers process data because they contain the specific instructions for what a computer or program does. **(ordered set of clearly defined instructions used to complete a task)**

## Method

A predefined set of instructions that can be used to accomplish a task. It is available on a specific object. For example, a turtle object has a list of methods that can be used to manipulate the object, such as forward(), backward(), and so on.





# Imports



[ ]

Imports are very useful to save time on your coding project and can help expand the code itself.

Some common imports to learn

- Turtle modules
- Random

Importing turtle will look like this

{ }

```
1 #Importing turtle module
2 import turtle as trtl
3 window = trtl.Screen()
4 window.mainloop()
```

Random is a collection of functions that can be used to randomize something as you can guess.

Importing random will look like this:  
More will be added in code but before you do that, you have to import it.

```
#Importing random
import random
```

{((({>>}))<<}

- [ ]



# Try It Yourself!



Try to start a turtle module like seen in the example, start a screen, and loop it. Once you have the basics, you can add things to have your trtl do.

## Some Ideas

You can do the following:

```
trtl.circle(insertsize)
```

```
trtl.backward(insertdistance)
```

```
trtl.forward(insertdistance)
```

```
trtl.pencolor("insertpythoncolorinquotes")
```

Once you have finished experimenting, I challenge you to attempt to make a square using `trtl.forward`, `trtl.left(angle)`, and `trtl.right(angle)`.

List of eligible turtle colors ↓

	aquamarine		blue2		chartreuse1		cyan		DarkOliveGreen3		DarkSlateGray
	aquamarine1		blue3		chartreuse2		cyan1		DarkOliveGreen4		DarkSlateGray1
	aquamarine2		blue4		chartreuse3		cyan2		DarkOrange		DarkSlateGray2
	aquamarine3		BlueViolet		chartreuse4		cyan3		DarkOrange1		DarkSlateGray3
	aquamarine4		brown		chocolate		cyan4		DarkOrange2		DarkSlateGray4
	azure		brown1		chocolate1		DarkBlue		DarkOrange3		DarkSlateGray
	azure1		brown2		chocolate2		DarkCyan		DarkOrange4		DarkTurquoise
	azure2		brown3		chocolate3		DarkGoldenrod		DarkOrchid		DarkViolet
	azure3		brown4		chocolate4		DarkGoldenrod1		DarkOrchid1		DeepPink
	azure4		burlywood		coral		DarkGoldenrod2		DarkOrchid2		DeepPink1
	beige		burlywood1		coral1		DarkGoldenrod3		DarkOrchid3		DeepPink2
	bisque		burlywood2		coral2		DarkGoldenrod4		DarkOrchid4		DeepPink3
	bisque1		burlywood3		coral3		DarkGray		DarkRed		DeepPink4
	bisque2		burlywood4		coral4		DarkGreen		DarkSalmon		DeepSkyBlue



# Operators

Operators are symbols or keywords that operate on one or more values(used in conditionals)

## Relational



Example showed  
on next slide

Used to compare two values and return a boolean result(true or false).

- `==` (Equal to)
- `!=` (Not equal to)
- `>` (Greater than)
- `<` (Less than)
- `>=` (Greater than or equal to)
- `<=` (Less than or equal to)

## Logical

Used to combine conditional statements.

- `and` (Returns `True` if both operands are `True`)
- `or` (Returns `True` if at least one operand is `True`)
- `not` (Reverses the logical state of its operand)

# What Are Conditionals?

{ } Conditionals are the “if... then” of programming. In your code if you wanted a result to rely on an action you would use a conditional. For example, if  $1 + 1 = 2$ , then set background color to pink. However, to code a conditional in Python you would use if and else. Formatting a conditional is very specific, you need to start with if and follow it with your variable and an operator as pictured. After those are established you need to put a colon to define the new code block. If you hit enter after putting the colon, it will automatically indent so you can properly put your condition. Then you follow the same format for “else”. ( )

```
test = 1
```

```
if test == 1:  
    wn.bgcolor("pink")  
else:  
    wn.bgcolor("blue")
```



This conditional turns the background color different depending on if the variable “test” equals 1. Since `test == 1` returns a true result the background color is set to pink.



# Let's Do It!

{ }

Using your knowledge of operators and conditionals, make a code segment using all.

For example here is what I wrote:

```
#Importing random
answer = int(input("Rate HackBI so far on a rate from 1-10?"))
if (answer < 10) or (answer < 1):
    print("Nooo")
else:
    print("Yay!!")
```

To explain my code, we are assigning the user's input to the question and changing it to a number so it can be comparable to other numbers. If the answer is less than one or ten, the program will print no. If it is anything else, it prints yay!

Now, try making something similar.

[ ]

# Loops

There are two common loops used in python: while and for. They can be used to repeat segments of code until a specific condition is met.

```
count = 0
while count < 5:
    print(count)
    count += 1
```

This is a while loop used to display numbers from 0-4.

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

Example of a for loop used to iterate through a list of elements to print each element.

See how much more efficient it was compared to printing each variable!

# Importance of Code Syntax

## Colon

Colons need to be directly placed after conditionals, loops, and when creating a function, or else the program will deem a syntax error and the program will malfunction.

## Spacing

Proper spacing between code segments are recommended to increase code's organization and readability.

## Indenting

For the next line after a colon is placed, and indent **NEEDS** to be placed. Sometimes it will be placed automatically, but be sure to check.

# Lists

Lists in Python are used to store ordered collections of items, which can be of any type (e.g., numbers, strings, objects). You can add, remove, or modify elements in a list, and you can also access them via indices (starting from 0). You can use the `import random` and assign it to your lists and the program will choose a random value from your list. For game aspect the `random.choice()` method can be used to randomly select an item from a list. If your lists contains words instead of numbers you need to have quotes around each value and comma after.

```
testlist = ["blue", "pink", "yellow"]
```

```
for step in testlist:  
    color = random.choice(testlist)  
    wn.bgcolor(color)
```



In this example a list is made with three values. I used a loop to access the list. I initiated “color” by assigning a random value using the `random` import from my list to it. Using the `color` variable in place of hardcoding, a color in the program takes a random color from my list and makes that the background color

# Functions AKA Procedure

## Procedure



A named group of programming instructions that may have parameters and return values. Procedures are referred to by different names, such as method or function, depending on the programming language.(AKA function)



In simpler terms, a function in Python is a block of reusable code that performs a specific task. Instead of writing the same code multiple times, you can define it once in a function and call it whenever you need it.



To initialize a function you define it starting with the **def** keyword.



Functions can take inputs, called *parameters*, which allow you to customize what the function does.

```
def spotsizes():  
    Size = rand.randint(1,10)  
    spot.shapesize(Size)
```



In this example I am defining the procedure “spotsizes” which assigns a random size between 1 and 10 to my spot. Since I made this into a procedure every time I want to make a new spot size I just call on “spotsizes()” instead of re-coding this whole thing. It may seem a little unimportant right now, but in longer programs making a procedure is a major time saver.

# Thanks!

Do you have any questions?

Feel free to ask now, or find us later!

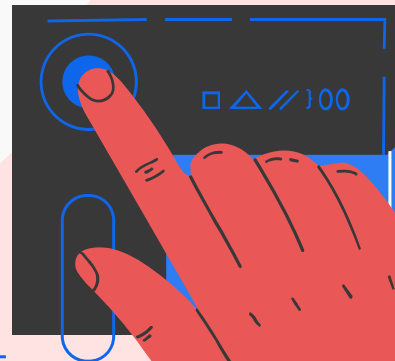
We are happy to help you find answers!

Thank you for taking the time to attend our workshop!

**CREDITS:** This presentation template was created by Slideshow, and includes icons by Flaticon, and infographics & images by Freepik



{{({{>>}})}<<}



(({{>>01□□□}}))

```
((: 00 - =>>})  
{ (<1 00 1 000 >>)}  
((: 0)>"< )  
<01 001} +100 0}>  
((: 0)>"< )  
{ (<1 00 1 000 >>)}
```

