Let’s Learn Python

The Computer Science Team

Do you know what Python is?

Type it into the chat!

TABLE OF CONTENTS

Panel 3

LEARNING THE BASICS

Comments:

→ Statements with the purpose of making code easier for humans to understand, and are ignored by the program & compiler

→ "#" The hashtag symbol will create a single line comment in Python

→ Python does not support multi - line comments (unlike languages such as Java)

LEARNING THE BASICS

Commenting Code Example:

→ To create multi - line comments you can use consecutive single line comments

```python
def my_function(a, b):
  a = 2a
  b = b**2
  c = (a+b)/a

  mylist = []
  count = 0

  while count != 10:
    mylist.append(c)

  return mylist
```

LEARNING THE BASICS

Print Statements:

- Statements used to print output to the terminal/console
- The function can be used by coding “print(Insert Value To Be Printed)”

Examples:

- print(“This will print on the console”)
- print(“Hello World!”)

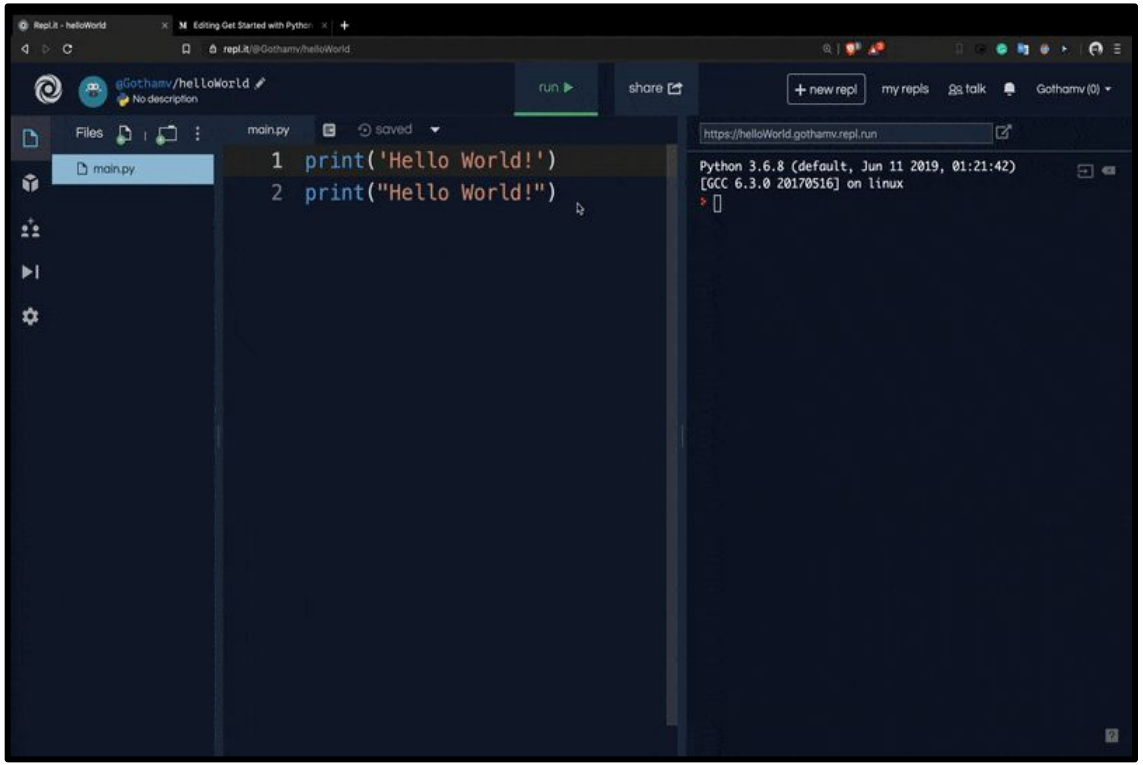
LEARNING THE BASICS

Printing Code Example:

→ Note: Unlike other programming languages Python does not require any syntax to end an instruction!

→ For Example:

Unlike Java there is no semicolon at the end of the statements



LEARNING THE BASICS

Gathering User Input:

- Used to gather input from the user in terminal
- Can be stored in a variable
- Simply set a variable as an **input(**prompt**) or raw_input()**

Examples:

```
name = input("enter your name: ")
print(name)
```

LEARNING THE BASICS

→ NOTE: There is no need to set a variable type when gathering input

→ Python automatically identifies whether user entered a string or a number or list. If the input provided is not correct then either syntax error or exception is raised.

Python 3

```
# Python program showing
# a use of input()

val = input("Enter your value: ")
print(val)
```

LEARNING VARIABLES

Variables:

- Containers for storing data values
- The main data types include: Text, Numeric, Boolean, and Sequence
- Text: String Variables
- Numeric: int, float, and complex Variables
- Boolean: bool Variables
- Sequence: list, range, and tuple Variables

LEARNING VARIABLES

Creating Variables:

→ In Python there is no command for declaring a variable, it is created the moment it is initialized

→ String variables can either be declared by single or double quotes (both work)

→ You can get the type of the variable using function “print(type(variable name))”

LEARNING VARIABLES

Creating Variables Example:

String Variables…

Number Variables…

Boolean Variables…

Sequence Variables…

Variables.py

```python
#String Variables (Single or Double Quotes Can Be Used
a = "John"
b = 'Jackie'

#Number Variables
c = 5 #int type variable
d = 5.0 #float type variable
e = 1j #complex type variable

#Boolean Variables (True or Flase Values)
f = True
g = False

#Sequence Variables
h = ["apple", "banana", "cherry"] #list type variable
i = ("apple", "banana", "cherry") #tuple type variable
j = range(7) #range type variable
```

LEARNING VARIABLES

Overwriting Variables:

→ Variables can be changed (in type and value) even after they have been set

Casting Variables:

→ You can use casting to specify the data type of a variable

Get The Type of The Variable:

→ You can get the data type of the variable with the “type()” function

LEARNING VARIABLES

Overwriting Variables Example:

→ Lines 1 - 8

Casting Variables Example:

→ Lines 10 - 13

Getting The Type of The Variable Example:

→ Lines 15 - 19

Variables.py

```
1   #Overwriting Variables
2   k = 5 #k is of type int
3   k = "Sally" #k is now of type str (String)
4   print(k) #prints Sally
5
6   l = 5
7   L = "Sally"
8   print(l) #prints 5 because this is not overwriting - variables are case sensitive
9
10  #Casting Variables
11  m = str(3) #l will be '3'
12  n = int(3) #m will be 3
13  o = float(3) #n will be 3.05
14
15  #Getting The Type of Variables
16  p = 5
17  q = "Jack"
18  print(type(p)) #prints that it is an integer type
19  print(type(q)) #prints that it is a str type
```

LEARNING DECISION STATEMENTS

Decision Statements:

→ Decision Statements are based on if a condition is met or not

For Example:

→ If the user clicks spacebar make the character jump

→ If the user holds the “w” key the character moves forward

→ Decisions like these are required everywhere in programming

→ They decide the direction of flow of program execution

LEARNING DECISION STATEMENTS

If Statements Example:

→ If statements take an expressions and checks if it evaluates to “True” then the block of code inside the if statement is executed. If the expression evaluates to “False” then the block of code is skipped

```
a = 20 ; b = 20
if ( a == b ):
  print( “a and b are equal”)
print(“If block ended”)
```

Output:

```
a and b are equal
If block ended
```

```
num = 5
if ( num >= 10):
  print(“num is greater than 10”)
print(“if block ended”)
```

Output:

```
If block ended
```

LEARNING DECISION STATEMENTS

If Else Statements Example:

→ Checks the expression and executes the if block when it is "True" otherwise it will execute the else block of code. The else block should be right after the if block and it is executed when the expression is "False"

Example 1:

```
number1 = 20 ; number2 = 30
if(number1 >= number2 ):
  print("number 1 is greater than number 2")
else:
  print("number 2 is greater than number 1")
```

Output:

```
number 2 is greater than number 1
```

LEARNING DECISION STATEMENTS

If Else Statements Example:

→ Only one else statement is followed by an if statement. If you use two else statements after an if statement, then you get the following error…

"Syntax Error"

```
if (5>10):
  print(5)
else:
  print(10)
else:
  print("End")
```

Output:

```
SyntaxError: invalid syntax
```

LEARNING DECISION STATEMENTS

If - Elif Ladder Example:

→ Serve the same purpose as the else - if statements in Java C and C++

→ Can make complex decision making statements

→ Checks multiple expressions and executes the code as soon as one of the conditions is met/"True"

Test.py

```
num = 10
if (num == 0):
    print("Number is Zero")

elif (num > 5):
    print("Number is greater than 5")

else:
    print("Number is smaller than 5")
```

LEARNING DECISION STATEMENTS

Nested Statements Example:

→ Statements within statements

→ Nested if statements is an if statement inside another if statement

→ Python allows any number of if statements inside the another

→ Useful to make series of decisions

```
num1 = int( input())
num2 = int( input())

if( num1>= num2):
    if(num1 == num2):
        print(f'{num1} and {num2} are equal')
    else:
        print(f'{num1} is greater than {num2}')
else:
    print(f'{num1} is smaller than {num2}')
```

Output 1:

```
10
20
10 is smaller than 20
```

Output 2:

```
5
5
5 and 5 are equal
```

LEARNING LOOPS

Python While **Loops:**

→ With the while loop we can execute a set of statements as long as a condition is true.

```
# Program to add natural
# numbers up to
# sum = 1+2+3+...+n

# To take input from the user,
# n = int(input("Enter n: "))

n = 10

# initialize sum and counter
sum = 0
i = 1

while i <= n:
    sum = sum + i
    i = i+1    # update counter

# print the sum
print("The sum is", sum)
```

LEARNING LOOPS

Python For **Loops:**

→ A for loop is used for iterating over a sequence

```
# Program to find the sum of all numbers stored in a list

# List of numbers
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]

# variable to store the sum
sum = 0

# iterate over the list
for val in numbers:
        sum = sum+val

print("The sum is", sum)
```

PERSONAL PRACTICE

Problem #1:

→ Ask the user for 2 numbers.

If number 1 > number 2 print “Number 1 is larger”

Else if number 1 = number 2 print “They are the same in value”

Else print “Number 1 is smaller”

PERSONAL PRACTICE

Problem #2:

→ Set a word in a variable.(ex. Your name)

Use one of the Python loops taught to print each letter in the word one by one. If 10 letters have been outputted then exit the program using **exit()**

The End

Any Questions?