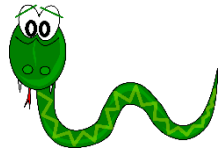
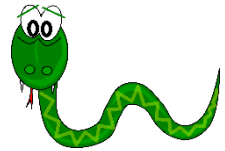


L#7. Introduction to python

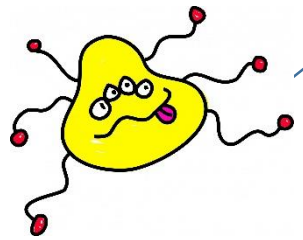
Part-1: Arithmetic Operators & Input/Output

Programming Languages



Programming languages are divided up into low-level languages and high-level languages. The closer the language is to machine language, the lower the level. High level languages make things easier to program.

Basic, FORTRAN, C/C++, PASCAL, Java, Python



what's python?

A low-level programming language for a computer or other programmable device specific to a particular computer architecture.

High-level Language

Easy to understand, use, portable, compiled or interpreted, less efficient

Assembly Language

Efficient, hard to use, machine dependent, not portable

Machine Language

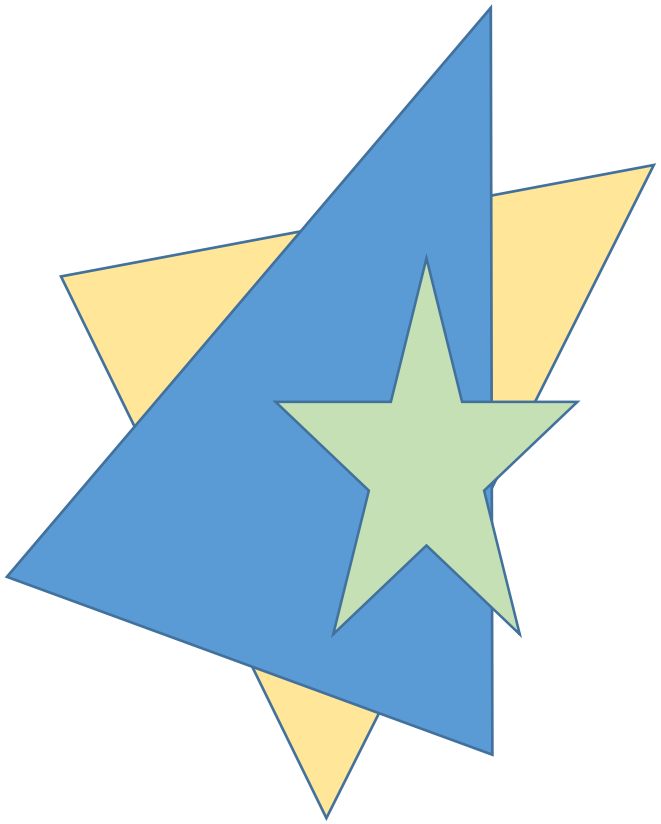
- Lowest-level language
- Binary-or hexadecimal encoded instructions that are directly executed by the hardware.
- Language is closely tied to the hardware design.
- Machine specific: machine language for one computer is different from that of a computer having a different hardware design.

Hardware

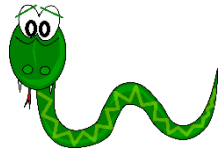
Is this a language?

Course Overview

- Basics:
  - Variables and constants
  - Arithmetic & Logical operators
  - Input/Output statements
- Programming Structures
  - Sequential
  - Selection: if
  - Repetition (also called Loops): while, for
- Arrays
  - Array Operators
- Functions
  - Library
  - User-defined functions



Python rules for constructing names

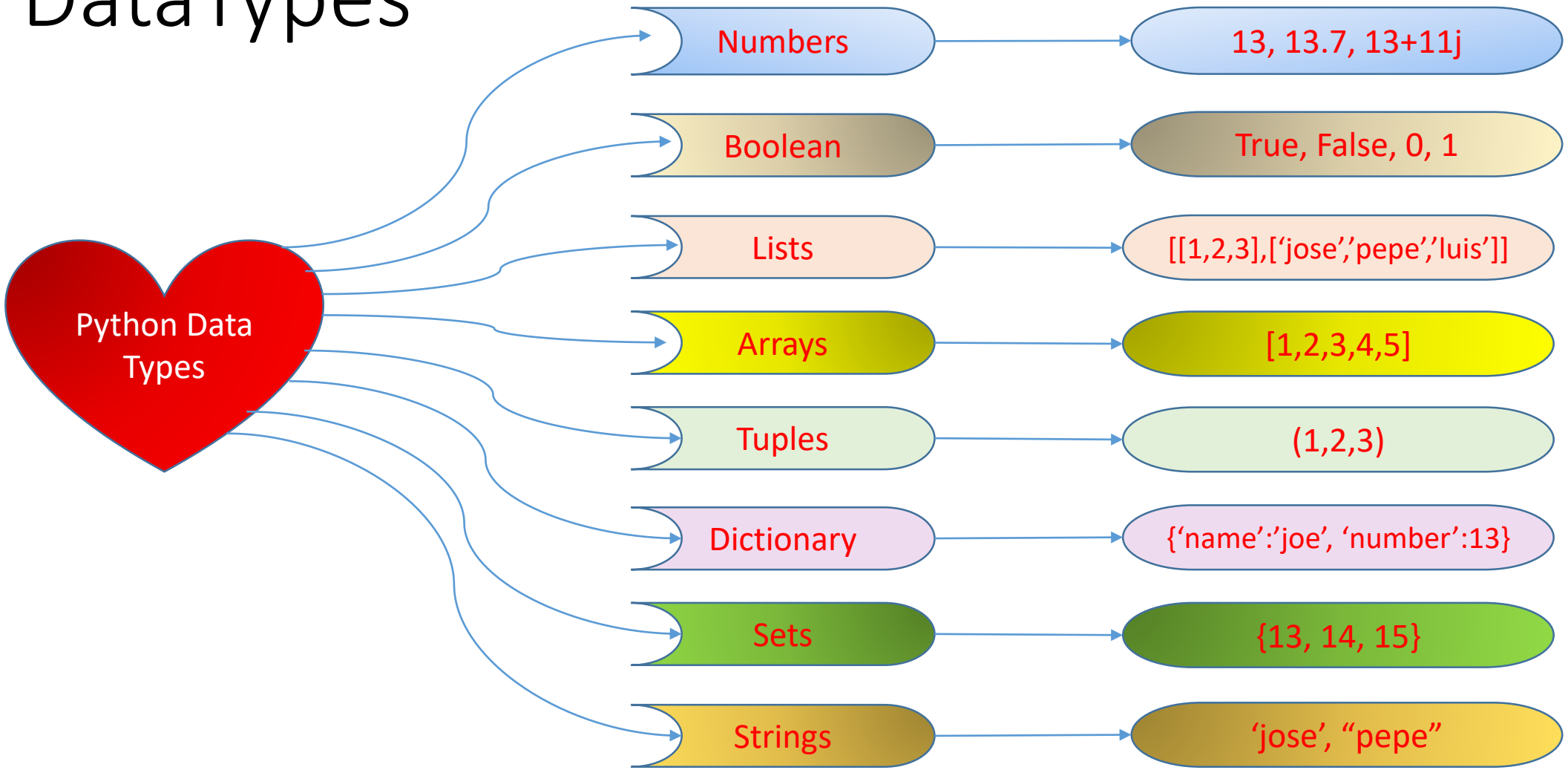


Apply to: variables, constants (parameters, e.g., Avogadro), functions, files names

- Must begin with a letter (a-z, A-Z, or underscore_)
- 0-9 (digits allowed but not accepted at start)
- Case sensitive (e.g., time≠TIME ≠Time)
- NO Blank Spaces
- Any reasonable length
- Avoid reserved words (keywords)
- (e.g., for, sin, while, if, etc.)

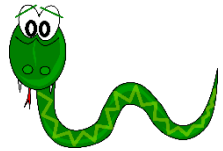
Right	Wrong
Vo, x1, y2, Ave,	Initial Velocity
quiz1, Quiz1, QUIZ1	7x
initialVelocity	$Amount
initial_Velocity	Quiz 1

DataTypes

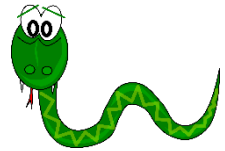


Adapted from:
http://www.regularpython.com/webscraper/python-basics/1/5/regularpython-regular-python-python-data-types

Arithmetic Operators in Python



Operator	*Meaning*	*Example*
+	Add two operands or unary plus	x + y +2
-	Subtract right operand from the left or unary minus	x - y -2
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	% (modulo) yields remainder from division of first argument by second. Numeric arguments are first converted to a common type.	x % y (remainder of x/y)
//	Integer Division yields integer quotients rounded to the left of the number line	x // y
**	Exponent - left operand raised to the power of right	x**y (i.e., x to the power y)



Examples

```
In [17]: 10//3
Out[17]: 3

In [18]: 10.0//3.0
Out[18]: 3.0
```

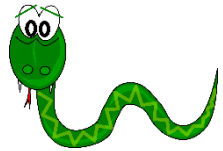
```
In [16]: -3**2
Out[16]: -9

In [17]: (-3)**2
Out[17]: 9
```

Since ** has higher precedence than unary -, then -3**2 will be interpreted as -(3**2) and thus result in -9. To avoid this and get 9, you can use (-3)**2.

```
In [14]: 10%3
Out[14]: 1

In [15]: 10.0%3.0
Out[15]: 1.0
```

```
In [1]: 10.0/3.0
Out[1]: 3.3333333333333335
```


OUTPUT: print function



Allow us to print numbers and explanatory text on the screen.

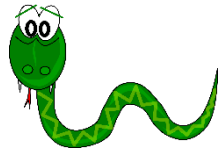
FILE: output01.py (also FILE: Python FORMATTED OUTPUT-Example-V3.docx

# Working with strings	Output
print('Hello There World!') # single quotes print("Hello There World!") # quotes print('''Hello There World!''') # triple single quotes print("""Hello There World!""") # triple quotes print('Hello Dear' + ' ' + 'Friends') # concatenation print('Hello','Dear','Friends') # concatenation	Hello There World! Hello There World! Hello There World! Hello There World! Hello Dear Friends Hello Dear Friends

import math # imports a python module
# most common is combination of text (string) + number 	
uno=1
dos=2
print('uno=', uno, 'dos=', dos, 'pi=', math.pi)
	uno= 1 dos= 2 pi= 3.141592653589793

Each print produces a new line. If you want to produce a blank line, use '\n' new line command

Print Commands



	print
Integer (int)	%d
Real (double precision)	%f, %e
Character (char) (one character)	%c
String	%s
newline	\n

```
# Print a 'salad bar' of data types
a = 3
b = 5.7
c = 1.7e-3
d = 'x'
e = 'Juan del Pueblo'
F=b>a

print('\n a= %d \n b= %3.1f \n c= %.2e \n d= %c \n e= %s \n' %(a,b,c,d,e))
print('F=%s',%(F))
```

OUTPUT

a= 3

b= 5.7

c= 1.70e-03

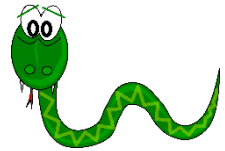
d= x

e= Juan del Pueblo

f=True

to print values in scientific notation

FILE: printCommands.py



INPUT

Allows entering data using the keyboard during program execution.

Syntax:

variable =input(“message”)

message to the user prompting for something, e.g., a value or string

the name of ONE variable, input returns strings..

name = input("Please enter your name: ")

Please enter your name: Juan del Pueblo

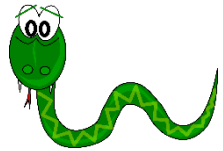
score = eval(input("Enter your final score: "))

Enter your final score: 98.5

Even if user entered a value in score, you would get back a string like ‘98.5’. The string can be converted into a number using an int() or a float() converter functions. Also eval() short for “evaluation” automatically converts the written input to the same type the user wrote on the screen. Use **eval()** when you plan to input numbers, for convenience.

score = eval(input("Enter your final score "))

THE END

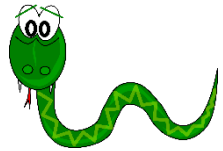


1. Built-in Functions
2. Math Functions
3. Numpy Functions
4. Statistics Functions
5. Matplotlib Functions

L#7. Introduction to python

Part-2: Built-in Functions, Math Functions

Python Built-in Functions



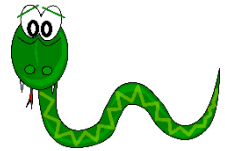
- Python has many functions available for use. Some don't require of importing modules. Don't require of dot notation. These functions are called built-in functions (also called intrinsic functions)
- In Python 3.8 (latest version), there are 68 built-in functions. Some of them are: abs(), input(), max(), min(), sum(), round(), print(), type(), etc.

Method	Description
abs()	returns absolute value of a number
all()	returns true when all elements in iterable is true
any()	Checks if any Element of an Iterable is True

```
# Example
N=-125; M=125; P=0
print("The absolute value of:")
print(N,"is",abs(N))
print(M,"is",abs(M))
print(P,"is",abs(P))
```

https://www.programiz.com/python-programming/methods/built-in
https://docs.python.org/3/library/functions.html

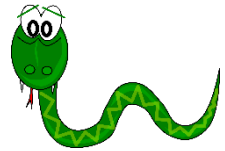
Modules & Dot Notation



The functionality in Python is provided by modules. The Python Standard Library is a large collection of modules that provides cross-platform implementations of common facilities, e.g., the math module. Below three ways to import a module: module.function()

import math **x = math.cos(2 * math.pi)** **print(x)**	• need to use prefix “math.” • includes the whole module and makes it available for use later in the program.
from math import cos, pi **x = cos(2 * pi)** **print(x)**	• import only a few selected functions from a module by explicitly listing which ones we want to import
from math import * **x = cos(2 * pi)** **print(x)**	• wild card * imports all symbols (functions and variables) in a module to the current namespace (so that we don't need to use the prefix “math." every time to call a function)

Math module: A Sample of Mathematical Functions

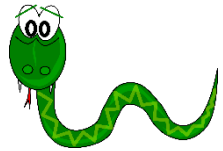


Function	Description
ceil(x)	Returns the smallest integer greater than or equal to x.
fabs(x)	Returns the absolute value of x
factorial(x)	Returns the factorial of x
floor(x)	Returns the largest integer less than or equal to x
fmod(x, y)	Returns the remainder when x is divided by y
fsum(iterable)	Returns an accurate floating point sum of values in the iterable
modf(x)	Returns the fractional and integer parts of x
trunc(x)	Returns the truncated integer value of x
exp(x)	Returns e**x
log(x[, base])	Returns the logarithm of x to the base (defaults to e)
log10(x)	Returns the base-10 logarithm of x
sqrt(x)	Returns the square root of x

Function	Description
acos(x)	Returns the arc cosine of x
cos(x)	Returns the cosine of x
sin(x)	Returns the sine of x
tan(x)	Returns the tangent of x
degrees(x)	Converts angle x from radians to degrees
radians(x)	Converts angle x from degrees to radians
acosh(x)	Returns the inverse hyperbolic cosine of x
cosh(x)	Returns the hyperbolic cosine of x
sinh(x)	Returns the hyperbolic cosine of x
tanh(x)	Returns the hyperbolic tangent of x
pi	Math constant, (3.14159...)
e	Math constant, (2.71828...)

THE END

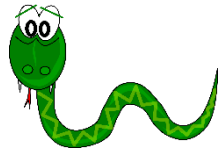
Take a look to this reference for an explanation of each mathematical function in the math module
References: https://docs.python.org/3/library/math.html



L#7. Introduction to python

Part-3: Selection Structure: If Statement

Python Selection Structure: Three versions



Allows the program flow to run into one of several choices

```
if  condition:
        statements
```

```
if  condition:
        statements-1
else:
        statements-2
```

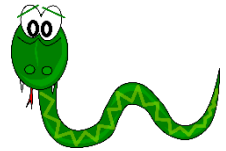
```
if  condition-1:
        stats-1
elif  condition-2:
        stats-2
        …
elif  condition-n:
        stats-n
else:   # optional
        stats-n+1
```

Nested structures

"elif" is one word (i.e., abbreviation of 'else if')
One "else" and its statements are optional

```
# Use CRTL+L to clear console screen
score=float(input("Enter your final score: "))

if score>=90:
    LG="A"
elif score>=80:
    LG="B"
elif score>=70:
    LG="C"
elif score>=60:
    LG="D"
else:
    LG="F"

print("You got ",LG)
```

The messy programmer

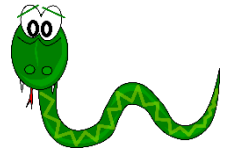


```
if score<60:
    LG="F"
elif score<70:
    LG="D"
elif score<80:
    LG="C"
elif score<90:
    LG="B"
else:
    LG="A"
```

```
if score<70 and score>=60:
    LG="D"
elif score<60:
    LG="F"
elif score<80 and score>=70:
    LG="C"
elif score>=90:
    LG="A"
else:
    LG="B"
```

The if statement examples

Boolean operators (Relational)

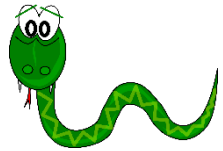


Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to

```
# Example:
a=5; b=7; c=9; d=11
X=a<b
Y=a<b and c<d
Z=not Y

print(a>b)      # False
print(a==b)     # False
print(a<=b)     # True
print(a>=b)     # False
print(a!=b)     # True
print(not(a!=b))    # False
print(X)        # True
print(Y)        # True
print(Z)        # False
print(1<a<6)    # True: 1<a and a<6
print(10<c<20)   # False 10<c and c<20
```

BooleanOperators.py

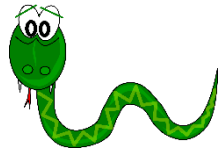
Logical operators:



Operator	Meaning
operand **and** *operand*	True if both operands are True
operand **or** *operand*	False if both operands are False
not *operand*	True if operand is False, False if operand is True

THE END

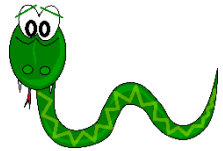
```
# Example
score=85
if score>=80 and score<90:
  print(score,"You got B")

AEE=input("Tiene ud luz?(YES or NO) ")
SOLARPANELS=input("Tien placas solares?(YES or NO) ")
if AEE=='YES' or SOLARPANELS=='YES':
  print("You are a lucky enligthened student")

healthy=eval(input("Are you sick?(True or False)"))
if not healthy:
  print("Ud es bendecido")
else:
  print("Sorry, cuidese mucho")
```



L#7. Introduction to python

Part-4: Repetition Structure: while and for loops

Python two repetition structures: Loops



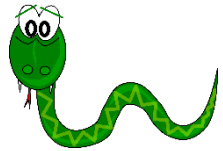
Two basic syntax forms:

```
for item in sequence:
    statements
```

```
while condition:
    statements
```

- Python's for statement iterates over the items, objects, values of any sequence (list, string, tuple, array), in the order that they appear.
- Items (objects) represent control variable, counter, running index
- Statements are indented (same indentation)
- To indent below each structure choose the tab key.

for Loop example & Table Construction(1)



Write a program to produce a table of x vs y, where x=[0,9] with step 1, and $y = x^2$:

```
import math
print('   x       y  ')
for x in [0,1,2,3,4,5,6,7,8,9]:
        y = math.sqrt(x)
        print('%.3f   %.3f   ' %( x, y))
```

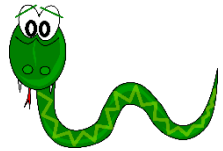
```
import math
print('   x       y  ')
for x in range(0,10,1):
        y = math.sqrt(x)
        print('%.3f   %.3f   ' %( x, y))
```

OUTPUT

x	y
0.000	0.000
1.000	1.000
2.000	1.414
3.000	1.732
4.000	2.000
5.000	2.236
6.000	2.449
7.000	2.646
8.000	2.828
9.000	3.000

range(start,stop,step)
range(0,10,1),
range(0,10), and
range(10)

All 3 above returns
[0,1,2,3,..,9]
stop is not included

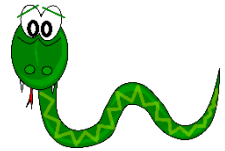


iterable, iterator

- **range()** produces the successive items of the desired sequence when you iterate over it, but it doesn't store the sequence, thus saving space. This function type is called an *iterable* object
- *iterable*, produces successive items until the supply is exhausted without storing them.
- An *iterator* is the manager of the *iterable*. The *for* statement is such an *iterator*. The function *list()* is another; both create lists from *iterables*:

```
print(list(range(10))) #range is the iterable; list is the iterator
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
# run instead of the above:  print(range(10))
                                # why the list is not generated?
```

iterable, iterator(2)



```
for city in ["Mayaguez", "San Juan", "Ponce", "Aibonito"]:
    print(city)
print()

# To stay in same line use end="somthing" argument:
for city in ["Mayaguez", "San Juan", "Ponce", "Aibonito"]:
    print(city,end = ", ")
print("\n")

for greens in ("Lettuce", "Kale", "Pepper","Spinach"):
    print(greens, end=" + ")
print("=Good Salad \n")

for char in "Iteration is extremely easy":
    print(char, end = " ")
```

OUTPUT

```
Mayaguez
San Juan
Ponce
Aibonito
```

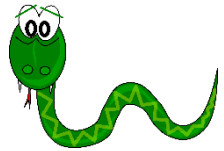
```
Mayaguez, San Juan, Ponce, Aibonito,
```

```
Lettuce + Kale + Pepper + Spinach + =Good Salad
```

```
I t e r a t i o n  i s  e x t r e m e l y  e a s y
```

(12)(17) List; (21) Tuple; (25) String. File loops02.py

while loop & Table Construction(2)



Write a program to produce a table of x vs y, where x=[0,9] step 1, and $y = \sqrt{x}$:

```
import math
x=0.0
print('%7s  %7s' %('x','y'))

while x<=9.0:
    y = math.sqrt(x)
    x+=1.0     # x = x +1
    print('%7.3f  %7.3f' %( x, y))
```

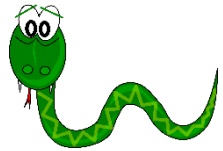
Compare to:

```
print('%7s  %7s' %('x','y'))
for x in range(0,10,1):
        y = math.sqrt(x)
        print('%7.3f   %7.3f ' %( x, y))
```

OUTPUT

x	y
0.000	0.000
1.000	1.000
2.000	1.414
3.000	1.732
4.000	2.000
5.000	2.236
6.000	2.449
7.000	2.646
8.000	2.828
9.000	3.000

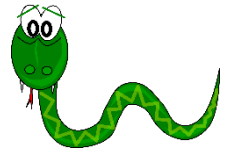
for Loop QUIZ



Write a **python** program to produce a well-formatted conversion table of Centigrades (C) on the range of [-250, 250, step 50], to degrees Fahrenheit, Rankine, and Kelvin degrees. Use a for loop.

Include code and output and submit team-work word-format report to Google Classroom. Before Xxday midnight, month/day/2020.

Quiz



$$\frac{1^2}{1}+\frac{2^2}{3}+\frac{3^2}{5}+\cdots+\frac{n^2}{2n-1}$$

Write a python code to solve the series up to n=10. Include code and output. Submit team-work word-format report to Google Classroom before midnight on month/day/year

THE END