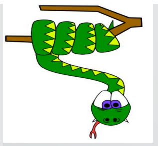
Cyber Girls

April 13, 2019



python™

Quick Tutorial

Naming Rules

- Names are case sensitive and cannot start with a number. They can contain letters, numbers, and underscores.

  `bob   Bob   _bob   _2_bob_   bob_2   BoB`

- There are some reserved words:

  `and, assert, break, class, continue, def, del, elif, else, except, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while`

Expressions

- **expression**: A data value or set of operations to compute a value.

  Examples: `1 + 4 * 3`

  `42`

- Arithmetic operators we will use:
  - `+ - * /` addition, subtraction/negation, multiplication, division
  - `%` modulus, a.k.a. remainder
  - `**` exponentiation

- **precedence**: Order in which operations are computed.
  - `* / % **` have a higher precedence than `+ -`

    `1 + 3 * 4` is `13`

  - Parentheses can be used to force a certain order of evaluation.

    `(1 + 3) * 4` is `16`

Integer division

- When we divide integers with / , the quotient is also an integer.

```
       3                       52
 4 )  14               27 )  1425
      12                     135
       2                       75
                               54
                               21
```

  - More examples:
    - `35 / 5` is `7`
    - `84 / 10` is `8`
    - `156 / 100` is `1`

- The % operator computes the remainder from a division of integers.

```
       3                  43
 4 )  14             5 )  218
      12                  20
       2                   18
                           15
                            3
```

Real numbers

- Python can also manipulate real numbers.
  - Examples: `6.022   -15.9997   42.0   2.143e17`
- The operators `+ - * / % ** ( )` all work for real numbers.
  - The / produces an exact answer: `15.0 / 2.0` is **`7.5`**
  - The same rules of precedence also apply to real numbers:
    Evaluate `( )` before `* / %` before `+ -`
- When integers and reals are mixed, the result is a real number.
  - Example: `1 / 2.0` is `0.5`
  - The conversion occurs on a per-operator basis.

```
7 / 3 * 1.2 + 3 / 2
  2   * 1.2 + 3 / 2
    2.4     + 3 / 2
    2.4     +   1
```

Math commands

Command name	Description
`abs(`***value***`)`	absolute value
`ceil(`***value***`)`	The smallest integer not less than **Value**
`cos(`***value***`)`	cosine, in radians
`floor(`***value***`)`	The largest integer not greater than **Value**
`log(`***value***`)`	logarithm, base *e*
`log10(`***value***`)`	logarithm, base 10
`max(`***value1, value2***`)`	larger of two values
`min(`***value1, value2***`)`	smaller of two values
`round(`***value***`)`	nearest whole number
`sin(`***value***`)`	sine, in radians
`sqrt(`***value***`)`	square root

Constant	Description
`e`	2.7182818...
`pi`	3.1415926...

Variables

- **variable**: A named piece of memory that can store a value.
  - Usage:
    - Compute an expression's result,
    - store that result into a variable,
    - and use that variable later in the program.

- **assignment statement**: Stores a value into a variable.
  - Syntax:

    name* = *value

  - Examples: `x = 5`
    `gpa = 3.14`

```
x  [ 5 ]        gpa  [ 3.14 ]
```

  - A variable that has been given a value can be used in expressions.
    `x + 4` is `9`

- `print` : Produces text output on the console.
- Syntax:

  print "***Message***"

  print ***Expression***

  - Prints the given text message or expression value on the console, and moves the cursor down to the next line.

  print ***Item1*** , ***Item2*** , ..., ***ItemN***

  - Prints several messages and/or expressions on the same line.
- Examples:

```
print "Hello, world!"
age = 45
print "You have", 65 - age, "years until retirement"
```

Output:

```
Hello, world!
```

- `input` : Reads a number from user input.
  - You can assign (store) the result of `input` into a variable.
  - Example:

```
age = input("How old are you? ")
print "Your age is", age
print "You have", 65 - age, "years until retirement"
```

Output:

```
How old are you? 53
Your age is 53
You have 12 years until retirement
```

- **for loop**: Repeats a set of statements over a group of values.
  - Syntax:

    for ***variableName*** in ***groupOfValues*** :

    statements

    - We indent the statements to be repeated with tabs or spaces.
    - ***variableName*** gives a name to each value, so you can refer to it in the ***statements***.
    - ***groupOfValues*** can be a range of integers, specified with the `range` function.
  - Example:

```
for x in range(1, 6):
    print x, "squared is", x * x
```

Output:

```
1 squared is 1
2 squared is 4
3 squared is 9
4 squared is 16
```

The `range` function specifies a range of integers:

- `range(`***start***, ***stop***`)` - the integers between ***start*** (inclusive) and ***stop*** (exclusive)
- It can also accept a third value specifying the change between values.
  - `range(`***start***, ***stop***, ***step***`)` - the integers between ***start*** (inclusive) and ***stop*** (exclusive) by ***step***
- Example:

```
for x in range(5, 0, -1):
    print x
print "Blastoff!"
```

Output:

```
5
4
3
2
1
```

Cumulative loops

- Some loops incrementally compute a value that is initialized outside the loop. This is sometimes called a *cumulative sum*.

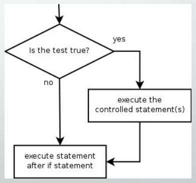
```
sum = 0
for i in range(1, 11):
    sum = sum + (i * i)
print "sum of first 10 squares is", sum

Output:
sum of first 10 squares is 385
```

if

- **`if` statement**: Executes a group of statements only if a certain condition is true. Otherwise, the statements are skipped.
  - Syntax:

```
if condition:
    statements
```

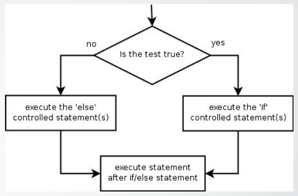

is the test true?
yes
no
execute the controlled statement(s)
execute statement after if statement

- Example:

```
gpa = 3.4
if gpa > 2.0:
    print "Your application is accepted."
```

- **`if/else` statement**: Executes one block of statements if a certain condition is True, and a second block of statements if it is False.
  - Syntax:

```
if condition:
    statements
else:
    statements
```

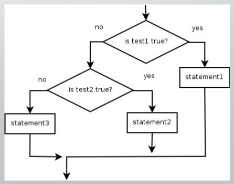

no
is the test true?
yes
execute the 'else' controlled statement(s)
execute the 'if' controlled statement(s)
execute statement after if/else statement

- Example:

```
gpa = 1.4
if gpa > 2.0:
    print "Welcome to Mars University!"
else:
    print "Your application is denied."
```

- Multiple conditions can be chained with `elif` ("else if"):

```
if condition:
    statements
elif condition:
    statements
else:
    statements
```



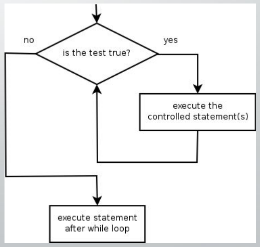
- **`while` loop**: Executes a group of statements as long as a condition is True.
  - good for *indefinite loops* (repeat an unknown number of times)

- Syntax:

```
while condition:
    statements
```

- Example:

```
number = 1
while number < 200:
    print number,
    number = number * 2
```


no
is the test true?
yes
execute the controlled statement(s)
execute statement after while loop

  - Output:

```
1 2 4 8 16 32 64 128
```

Logic

- Many logical expressions use *relational operators*:

Operator	Meaning	Example	Result
`==`	equals	`1 + 1 == 2`	`True`
`!=`	does not equal	`3.2 != 2.5`	`True`
`<`	less than	`10 < 5`	`False`
`>`	greater than	`10 > 5`	`True`
`<=`	less than or equal to	`126 <= 100`	`False`
`>=`	greater than or equal to	`5.0 >= 5.0`	`True`

- Logical expressions can be combined with *logical operators*:

Operator	Example	Result
`and`	`9 != 6 and 2 < 3`	`True`
`or`	`2 == 3 or -1 < 5`	`True`
`not`	`not 7 > 0`	`False`

- **string**: A sequence of text characters in a program.
  - Strings start and end with quotation mark `"` or apostrophe `'` characters.
  - Examples:

```
"hello"
"This is a string"
"This, too, is a string.   It can be very long!"
```

- A string may not span across multiple lines or contain a " character.

```
"This is not
a legal String."

 "This is not a "legal" String either."
```

- A string can represent characters by preceding them with a backslash.

`\t`	tab character
`\n`	new line character
`\"`	quotation mark character
`\\`	backslash character

  - Example: `"Hello\tthere\nHow are you?"`

- Characters in a string are numbered with *indexes* starting at 0:
  - Example:

```
name = "P. Diddy"
```

index	0	1	2	3	4	5	6	7
character	P	.		D	i	d	d	y

- Accessing an individual character of a string:

  variableName [***index***]

  - Example:

```
print name, "starts with", name[0]
```

Output:

```
P. Diddy starts with P
```

String properties

- `len(`***string***`)` - number of characters in a string (including spaces)
- `str.lower(`***string***`)` - lowercase version of a string
- `str.upper(`***string***`)` - uppercase version of a string

- Example:

```
name = "Martin Douglas Stepp"
length = len(name)
big_name = str.upper(name)
print big_name, "has", length, "characters"
```

Output:

```
MARTIN DOUGLAS STEPP has 20 characters
```

`raw_input` : Reads a string of text from user input.

- Example:

```
 name = raw_input("Howdy, pardner.
What's yer name? ")
 print name, "... what a silly name!"
```

 Output:

```
 Howdy, pardner. What's your name?
Paris Hilton
 Paris Hilton ... what a silly name!
```

text processing: Examining, editing, formatting text.

- often uses loops that examine the characters of a string one by one

- A `for` loop can examine each character in a string in sequence.
  - Example:

```
for c in "booyah":
    print c
```

Output:

```
b
o
o
y
a
h
```

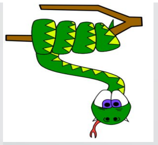
Strings and numbers

- `ord(`***text***`)` - converts a string into a number.
  - Example: `ord("a")` is `97`, `ord("b")` is `98`, ...
  - Characters map to numbers using standardized mappings such as *ASCII* (http://www.asciitable.com/)and *Unicode*.

- `chr(`***number***`)` - converts a number into a string.
  - Example: `chr(99)` is `"c"`

- **Example:** A program that performs a rotation cypher.
  - e.g. `"attack"` when rotated by 1 becomes `"buubdl"`

```
Hint: Look at the the ASCII table, a = 97, if rotated by 1,it becomes
98. 98 is for b. use for loop, ord(), and chr()
```



python™

Let's Code

Online Python Tools

Optional

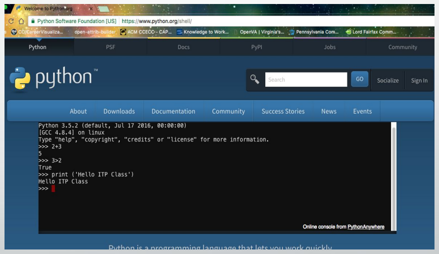
- 1. https:// www.tutorialspoint.com/execute_python3_online.php
- 2. https:// repl.it/languages/python3

- IDLE:
  - http :// rextester.com/l/python3_online_compiler
- Shell:
  - https://www.python.org/shell /

Shell

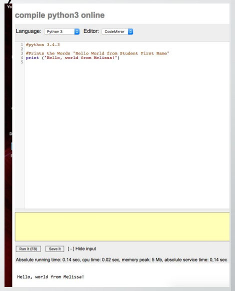
Go to Tab with Shell: https://www.python.org/shell/
Type the following into the Shell
Lines with >>> are the commands you type, others are results

>>> 2+3
5
>>>3>2
True
>>> print (‘Hello World’)
Hello World



```
Python 3.5.2 (default, Jul 17 2016, 00:00:00)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 2+3
5
>>> 3>2
True
>>> print ('Hello ITP Class')
Hello ITP Class
>>>
```

Integrated DeveLopment Environment (IDE/IDLE)



```
#python 3.4.3

#Prints the Words "Hello World from Student First Name"
print ('Hello, world from Melissa!")
```

Absolute running time: 0.14 sec, cpu time: 0.02 sec, memory peak: 5 Mb, absolute service time: 0,14 sec

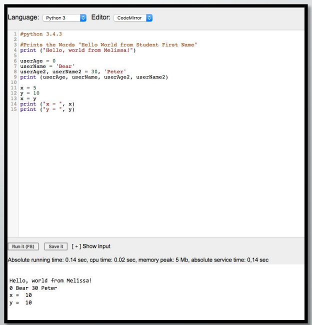
```
Hello, world from Melissa!
```

Open IDE tab: http://rextester.com/l/python3_online_compiler

1. Open Input Window, insert blank line
2. On line 3 Type:
   #Prints the Words “Hello World from Student First Name”
3. Change **print** statement to have the text ’from yourfirstname’ before exclamation point (!).
   Change yourfirstname to your real first name
4. Run It

Variable & Operators

- **Type in the IDE/IDLE**
  - **userAge = 0**
  - **userName = ‘Bear’**
  - **userAge2, userName2 = 30, ‘Peter’**
  - **print (userAge, userName, userAge2, userName2)**
  - **x=5**
  - **y=10**
  - **x=y**
  - **print (“x = “, x)**
  - **print (”y = “, y)**
  - **RUN IT**



```
#python 3.4.3

#Prints the Words "Hello World from Student First Name"
print ("Hello, world from Melissa!")

userAge = 0
userName = 'Bear'
userAge2, userName2 = 30, 'Peter'
print (userAge, userName, userAge2, userName2)

x = 5
y = 10
x = y
print ("x = ", x)
print ("y = ", y)
```

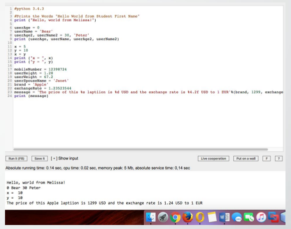
Absolute running time: 0.14 sec, cpu time: 0.02 sec, memory peak: 5 Mb, absolute service time: 0,14 sec

```
Hello, world from Melissa!
0 Bear 30 Peter
x =  10
y =  10
```

Data Types

In IDE/IDLE type the following:

- mobileNumber = 12398724
- userHeight = 1.28
- userWeight = 67.2
- userSpouseName = ‘Janet’
- brand = ‘Apple’
- exchangeRate = 1.23523544
- message = ‘The price of this %s laptop is %d USD and the exchange rate is %4.2f USD to 1 EUR’ %(brand, 1299, exchangeRate)
- print (message)



```
#python 3.4.3

#Prints the Words "Hello World from Student First Name"
print ("Hello, world from Melissa!")

userAge = 0
userName = 'Bear'
userAge2, userName2 = 30, 'Peter'
print (userAge, userName, userAge2, userName2)

x = 5
y = 10
x = y
print ("x = ", x)
print ("y = ", y)

mobileNumber = 12398724
userHeight = 1.28
userWeight = 67.2
userSpouseName = 'Janet'
brand = 'Apple'
exchangeRate = 1.23523544
message = 'The price of this %s laptiion is %d USD and the exchange rate is %4.2f USD to 1 EUR'%(brand, 1299, exchange
print (message)
```

Absolute running time: 0.14 sec, cpu time: 0.02 sec, memory peak: 5 Mb, absolute service time: 0,14 sec

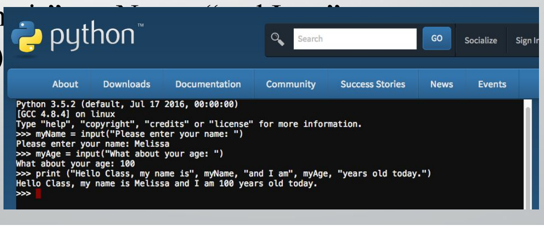
```
Hello, world from Melissa!
0 Bear 30 Peter
x =  10
y =  10
The price of this Apple laptiion is 1299 USD and the exchange rate is 1.24 USD to 1 EUR
```

Run It

SHELL - Interactivity

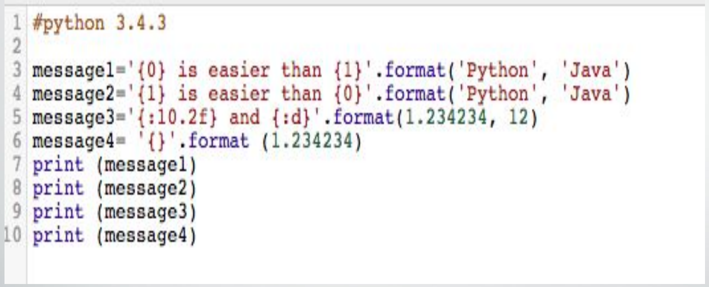
In the SHELL type the following:

- myName = input("Please enter your name: “)
  - **Answer Name**
- myAge = input ("What about your age: ")
  - **Answer Age**
- Print (“Hello Class, my nam
  myAge, “years old today.”)



```
Python 3.5.2 (default, Jul 17 2016, 00:00:00)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> myName = input("Please enter your name: ")
Please enter your name: Melissa
>>> myAge = input("What about your age: ")
What about your age: 100
>>> print ("Hello Class, my name is", myName, "and I am", myAge, "years old today.")
Hello Class, my name is Melissa and I am 100 years old today.
>>>
```

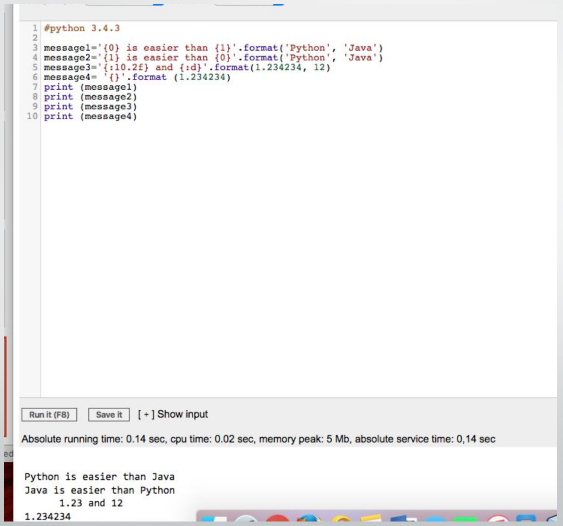
In IDLE/IDE



```
#python 3.4.3

message1='{0} is easier than {1}'.format('Python', 'Java')
message2='{1} is easier than {0}'.format('Python', 'Java')
message3='{:10.2f} and {:d}'.format(1.234234, 12)
message4= '{}'.format (1.234234)
print (message1)
print (message2)
print (message3)
print (message4)
```

Run it



```
#python 3.4.3

message1='{0} is easier than {1}'.format('Python', 'Java')
message2='{1} is easier than {0}'.format('Python', 'Java')
message3='{:10.2f} and {:d}'.format(1.234234, 12)
message4= '{}'.format (1.234234)
print (message1)
print (message2)
print (message3)
print (message4)
```

Absolute running time: 0.14 sec, cpu time: 0.02 sec, memory peak: 5 Mb, absolute service time: 0,14 sec

```
Python is easier than Java
Java is easier than Python
      1.23 and 12
1.234234
```