Computing Studies

Software Design and Development

Python 3.3 Syntax Reference

Buckhaven High School

Version 1

Contents

How to use this booklet

Even professional programmers look up examples of other code to help them write programs.

This booklet contains dozens of examples of Python 3.3 syntax.

If you can't remember how to write a particular statement or line of Python code, use the index at the top of this page and look up some examples. You should then adapt the example for the program you are currently writing.

Written by Mr G. Reid, PTC Technologies, Buckhaven High School - Nov 2013

This booklet has been written to aid covering the following content in the Scottish, National 4 and National 5 Computing courses but it may be used as a reference for anyone learning to write Python 3.3 programs.

	National 4	National 5
Computational Constructs	Exemplification and implementation of the following constructs: • expressions to assign values to variables • expressions to return values using arithmetic operations (+,-,*,/,^) • execution of lines of code in sequence demonstrating input - process - output • use of selection constructs including simple conditional statements • iteration and repetition using fixed and conditional loops	Exemplification and implementation of the following constructs: • expressions to assign values to variables • expressions to return values using arithmetic operations (+,-,*,/,^,mod) • expressions to concatenate strings and arrays using the operator • use of selection constructs including simple and complex conditional statements and logical operators • iteration and repetition using fixed and conditional loops • pre-determined functions (with parameters
Data Types and Structures	string numeric (integer) variables graphical objects	string, character numeric (integer and real) boolean variables 1-D arrays
Algorithm Specification		Exemplification and implementation of algorithms including • input validation

Formatting rules of Python

1. **Syntax (Keywords)**
   All Python 3.3 command words should be typed in lower case.
   input
   print
   while
   Most editors will highlight command words automatically in a different colour as you enter them.

2. **Variable Names**
   Variable names can only be single word. If you wish to use two words use a capital letter at the start of the second to make it stand out. (This is not actually required by Python but it will make your code more readable.)
   firstName
   bestScore
   telephoneNumber

3. **Indentation**
   Constructs that are more than one line long use indentation to note where the code for the construct finishes. The examples below show constructs that have 1,2 or 3 associated lines of code.

```
while answer == "N":
        answer = input("Would you like some advice")


if  answer == "Y":
        print("Always get out of bed on the right side")
        print("The other grass is not always greener")


for counter in range(6):
        print(name[counter])
        print("scored")
        print(points[counter])
print ("Finished")
```

} these three indented lines are all part of the for loop

} this is the next line of the program

Written by Mr G. Reid, PTC Technologies, Buckhaven High School - Nov 2013

Assigning Values to Variables and Arrays

=
[ ]

Variables can be assigned values as shown below.

Numeric assignment examples:


```
number = 973
diameter = 23.56
```

String assignment examples:


```
firstName = "Greg"
fullName = "Greg Reid"
```

Boolean assignment examples:


```
flag = False
test = True
```

Array assignment examples:

storing initial values in an array:

```
teamAsScore = [0]*10                #fill a 10 element array with zeros
or
teamAsScore = [3,5,2,3,4,5,2,3,5,5] #fill the array with selected values

teamNames = [""]*10                 #fill a 10 element array with null strings
or
teamNames = ["Buckhaven FC", "Methil United","Leven Academicals"]
```

assigning values to individual array elements:

```
speed[1] = 70
names[3] = "Gillian"
```

assigning values to an array using a for loop:

```
for num in range(5):
    tempertures[num] = 0
```

Using Input to Assign Values to Variables

input( )
int( )
float( )

An input statement can be used to assign (store) a value directly in a variable.

Examples using a variety of data types are shown below.

String assignment example:

```
firstName = input("Please enter your first name")
```

Numeric assignment examples:

to input an integer (whole number) the int( ) function is place round the input statement

```
number = int(input("please enter a whole number"))
```

to input a real number (one with decimal places) the float( ) function is place round the input statement

```
number = float(input("please enter a whole number"))
```

Written by Mr G. Reid, PTC Technologies, Buckhaven High School - Nov 2013

Using Print to Output Data and Variables

print()
,
“ ”

A print statement is used to display data or variables in an output window.

Print statements can be formatted in a variety of ways.

A few simple and more complex examples are shown below.

Simple print( ) examples:

```
print(“This is some text”)
```
- displaying text

```
dogName = “Penny”
print(dogName)
```
- displaying a string variable

```
shoeSize = 10
print(shoeSize)
```
- displaying a numeric variable

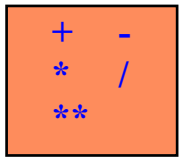
Combination print( ) examples (using commas):

```
dogName = “Penny”
print(“The dog’s name is “, dogName)
```
- text then variable

```
coins = 10
print(“You have”, coins, “50p pieces”)
```
- text, variable, text

```
name = “Arthur”
height = 175
print(name,  “is”, height, “cm tall”)
```
- variable, text, variable, text

Arithmetic Operations



Python allows arithmetic operations as part of an assignment to a variable or as part of an output statement.

Examples of both are shown below.

Arithmetic operations using **values (number)**:

```
AgeMonths = 34 * 12
```

N4

```
cost = 12 + 3 + 3.5
```

N4

```
average = (12 + 4 + 5 + 28 + 92) / 5
```

N4

```
cylinderVolume = (3.14 * 3**2) * 12
```

- note 3**3 = 3^3

N5

Arithmetic operations using **variables**:

```
totalCost = itemCost + itemVat
```

N4

```
years = months / 12
```

N4

```
cubeVolume = length**3
```

N5

Worked example

```
income = float(input("Please enter the income figure"))
expenditure = float(input("Please enter the expenditure figure"))
profit = income - expenditure
print("The profit is ",profit)
```

N4

Concatenating Strings

+
str( )

Concatenation means to join two or more strings (text) together into a single string.

This is done in Python using a + symbol.

Text examples:

```
fullName = “David” + “Mitchell”
```

```
print(“David” + “Mitchell”)
```

Text and variable example:

```
name = “Greg Reid”
print (“Welcome” + name)
```

Variable and variable example:

```
stockName = “ipad”
stockProduct = “32 Gb Air, Silver”
print(stockName + stockProduct)
```

To concatenate a string and a numeric value the numeric value must firat be converted to a string using the str( ) function.

```
age = 42
name = “Susie”
print(“Hello” + name + str(age))
```

Selection with Simple Conditions

if	==
>	<
>=	<=
!=	

Selection (or making decisions) involves the use of 'if' statements and conditions with operators.

Note that an 'if' statement is always followed by a colon ':' and lines of code associated with the if statement.

Examples of if statements with operators are shown below.

Equal To (==) example:

```
if size == 10:
    print("Correct size reached")
```

Greater Than (>) example:

```
if size > 10:
    print("Selected size too large")
```

Less Than (<) example:

```
if size < 10:
    print("An increase in size still required")
```

Greater Than or Equal To (>=) example:

```
if score >= 10:
    print("Well Done")
```

Less Than or Equal To (<=) example:

```
if size <= 10:
    print("Target not reached")
```

Not Equal To (!=) example:

```
if guess != 10:
    print("Sorry, the correct answer was 10")
```

Written by Mr G. Reid, PTC Technologies, Buckhaven High School - Nov 2013

Selection Options

else
elif

Several decisions can be combined in one by extending the 'if' statement using 'else' and 'elif'.

Examples of both are shown below.

Else example:

```
if score >=10:
    total = total + score*2
else:
    total = total - 2
```

Elif example:

```
if score > 20:
    total = total + score*3
elif score > 10:
    total = total + score*3
else:
    total = total - 2
```

Selection with Complex Conditions

and
or
not

'if' statements may also have combined (or complex) conditions.
Statements are combined using:

and - both conditions must be true
or - either condition may be true
not - opposite of conditions must be true

Examples of all three are shown below.

Complex condition using **and**:


```
if testMark >=10 and testMark <=20:
        print("You passed with a score of between 10 and 20")
else:
        print("Sorry, your mark was not between 10 and 20."
        print("You failed")
        print("Try again")
```

Complex condition using **or**:


```
if testMark <0 or testMark >20:
        print("Invalid test mark, please enter again")
        print("Mark must be in the range - 0 to 20")
else:
        print("Valid mark between 0 and 20 entered")
```

Complex condition using **not**:


```
if not(guess==10):
        print("Sorry, you guessed incorrectly")
elif guess>10
        print("Too high")
elif guess <10
        print("Too low")
else:
        print("Well Done, that answer was 10")
```

Unconditional Repetition

for range()

An unconditional loop is used to repeat lines of code a set number of times.

A variable is required in the for statement to keep count.

```
for counter in range(100)
```

Examples of 'for' loops are shown below.

Simple counting loop:

```
for counter in range(10):                #count 10 times from 0 to 9
    cost = float(input("Please enter the next cost"))
    total = total + cost
```

Simple counting loop using the loop variable:

```
for counter in range(0,20,4):            #count from 0 to 19 in steps of 4
    print(counter)                       #output is - 0,4,8,12,16
```

Simple loop using the loop variable with an array:

```
name = [""]*6
for counter in range(6):                 #count from 0 to 5
    print("Please enter name",counter)   #output using loop variable
    name[counter] = input( )             #input a string to the array
for counter in range(6):                 #count from 0 to 5 again
    print(name[counter])                 #display the array
```

Conditional Repetition

While

A loop that repeats until a set of conditions are true is called a conditional loop. Python uses the 'while' command for conditional loops. The statements inside the loop are indented.

As with 'if' statement a single simple conditional or multiple complex conditions may be used.

Examples of both are shown below.

Simple conditional repetition:

```
totalCost = 0
cost = 0
while totalCost < 100:
      cost = float(input("Enter the price of the next item"))
      totalCost = totalCost + cost
print("You have now exceeded £100)
print("
```

N4

Complex conditional repetition:

```
temperature = 0
while temperature >= 15 and temperature <= 25:
    temperature = float(input("Enter the current temperature"))
print("Alert - Room now outside acceptable temperature")
```

N5

Input Validation

'While' loops are used to ensure that values entered into a program are acceptable.

```
choice = ""
while choice != "Y" and choice != "N":
    choice = input("Do you wish to continue? Y/N")
```

N5

```
age = 0
while age <=0:
    age = int(input("Please enter your age"))
```

N4

Written by Mr G. Reid, PTC Technologies, Buckhaven High School - Nov 2013

Predetermined Functions with Parameters

Every programming language has predetermined (built-in) functions to carry out common tasks.

Page 8 used the predetermined function str( ) which is used to convert a numeric value to a string.

Examples of some other commonly used Python functions are shown below.

%
len( )
lower( )
upper( )
ord( )
chr( )
round( )
ramdom( )
randint( )

Mod function - % - returns the remainder when one number is divided by another

```
print(12%5)                 #this would display 2
                             (12/5, remainder 2)
```

Length function - len( ) - returns the number of characters in a string

```
print(len("Computing"))     #this would display 9
                             (9 letters in 'Computing')
```

Lower Case function - lower( ) - changes upper case letters in strings to lower case

```
sentence = "I NEVER could get the hang of Thursdays"
print(sentence.lower())
#this would display: i never could get the hang of thursdays
```

Upper Case function - upper( ) - changes lower case letters in strings to upper case

```
sentence = "I NEVER could get the hang of Thursdays"
print(sentence.upper())
#this would display: I NEVER COULD GET THE HANG OF THURSDAYS
```

Ord function - ord( ) - converts a single character to its ascii code equivalent

```
letter = "s"
print(ord(letter))          #this would display 115 (ascii value of 's')
```

Chr function - chr( ) - converts a single number to its ascii character equivalent

```
number = 115
print(chr(number))          #this would display s
```

Round function - round( ) - rounds a real number to a specified number of places

```
number = 12.3567
print(round(number,2))      #this would display 12.36
```

Written by Mr G. Reid, PTC Technologies, Buckhaven High School - Nov 2013

Random function - random( ) - generates a random real number between 0 and 1

```
import random                    #this function requires the random module
num = random.random()
print(num)
#this would display a random number in the form: 0.3424391381916535
```

Random Integer function - randint( ) - generates a random integer in a range

```
import random                    #this function requires the random module
num = random.randint(1,10)       #generate a random integer in the given range
print(num)
#this would display a integer between 1 and 10: for example 7
```

More Predetermined Functions - Beyond National 5

The use of functions is vital to becoming a good programmer. Here are some other useful functions that will not be covered in clas

Count function - count( ) - the number of times one string occurs in another string

```
sentence = "Round the rugged rock the ragged rascal ran"
print(sentence.count("r"))
#this would display 5 - the number of lower case r's found in the initial string
```

Split function - split( ) - splits up a string and create an array from the string

```
sentence = "Doctor Who the Musical"
words = sentence.split( )
#this would create an array with four elements
["Doctor","Who","the","Musical"]
Note - if the split brackets are left blank the split takes place at each space.
```

Find function - find( ) - returns the position of one string within another

```
sentence = "Footballers practice football skills during every football game"
print(sentence.find("football"))
#this would display 21 - the position in the string of the first 'football'
```

Replace function - replace( ) - finds a string and replaces it with another

```
sentence = "Davie Celtic obviously supports Celtic FC"
sentence = sentence.replace("Celtic","Rangers",2)
print(sentence)
#stores a sentence and replaces the first 2 occurrences of Celtic with Rangers
#the output would be:  Davie Rangers obviously supports Rangers FC
```


Written by Mr G. Reid, PTC Technologies, Buckhaven High School - Nov 2013