UNIVERSITY OF GLOUCESTERSHIRE

CT4031 – MATHS FOR DATA SCIENCE

LET'S START A PYTHON RECAP!

print

`print` : Produces text output on the console.

Syntax:

```
print ("Message")
```

Examples:

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

Output:

```
Hello, world!
You have 20 years until retirement
```

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 - int(age), "years until retirement")
```

Output:

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

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 - int(age), "years until retirement")
```

Output:

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

`input`

Exercise:

Write a Python program that prompts the user for his/her amount of money, then reports how many Nintendo Switch the person can afford, and how much more money he/she will need to afford an additional Switch.

Expressions

Expression: A data value or set of operations to compute a value.

Examples: `1 + 4 * 3`

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`

Math commands

Command name	Description
`abs(`***value***`)`	absolute value
`ceil(`***value***`)`	rounds up
`cos(`***value***`)`	cosine, in radians
`floor(`***value***`)`	rounds down
`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...

Python has useful commands for performing calculations.

To use many of these commands, you must write the following at the top of your Python program:

```
from math import *
```

Logical expressions

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*:

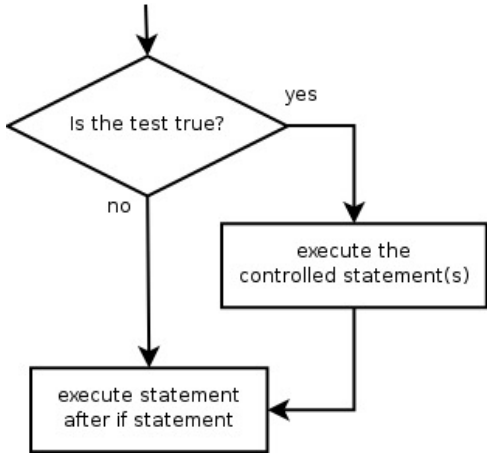
Operator	Example	Result
and	`9 != 6 and 2 < 3`	`True`
or	`2 == 3 or -1 < 5`	`True`
not	`not 7 > 0`	`False`

if/else

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

Syntax:

```
if condition :
    statements
```



Example:

```
Grade = 34
if Grade > 39:
    print ("You have passed.")
else:
    print ("You have failed.")
```

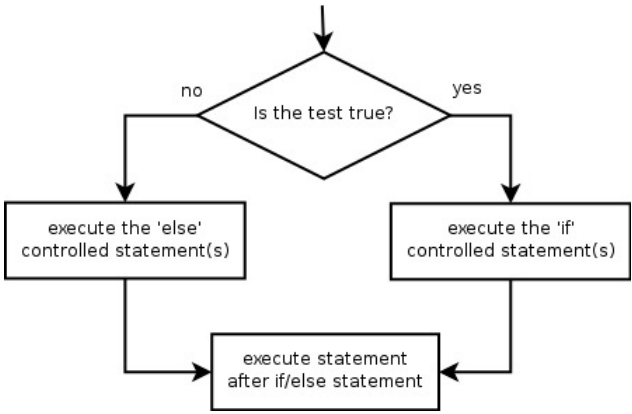
if/else

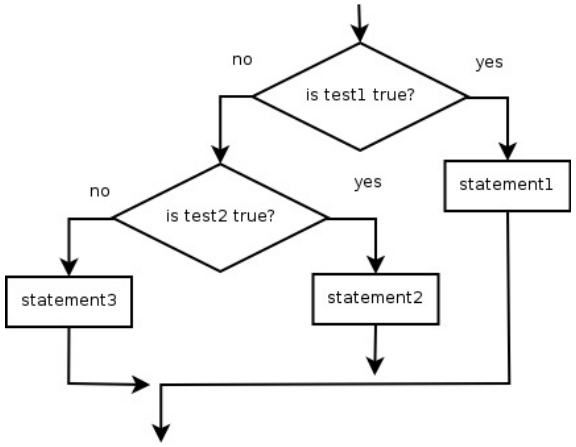
Multiple conditions can be chained with `elif` ("else if"):

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

Example:

```
Grade = 34
if Grade > 39:
    print ("You have passed.")
elif Grade > 0:
    print ("You have to do a reassessment.")
else:
    print("You have failed.")
```





Task - High/Low game

1) A player is shown a random number from 1 to 10, then asked to decide whether the next number will be ‘higher’ or ‘lower’. If the guess is correct the player is awarded with points

2) Extend, so that the player can place a bet (based on their ‘points’) prior to making a high-low guess.

The `for` loop

`for` loop: Repeats a set of statements over a group of values.

Syntax:

```
for variableName in groupOfValues:
    statements
```

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
5 squared is 25
```

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
```

Exercise: Write a Python program that computes the factorial of an integer.