Lecture 3: Conditionals

COMP90059 Introduction to Programming

Wally Smith

School of Computing & Information Systems

Lecture Overview

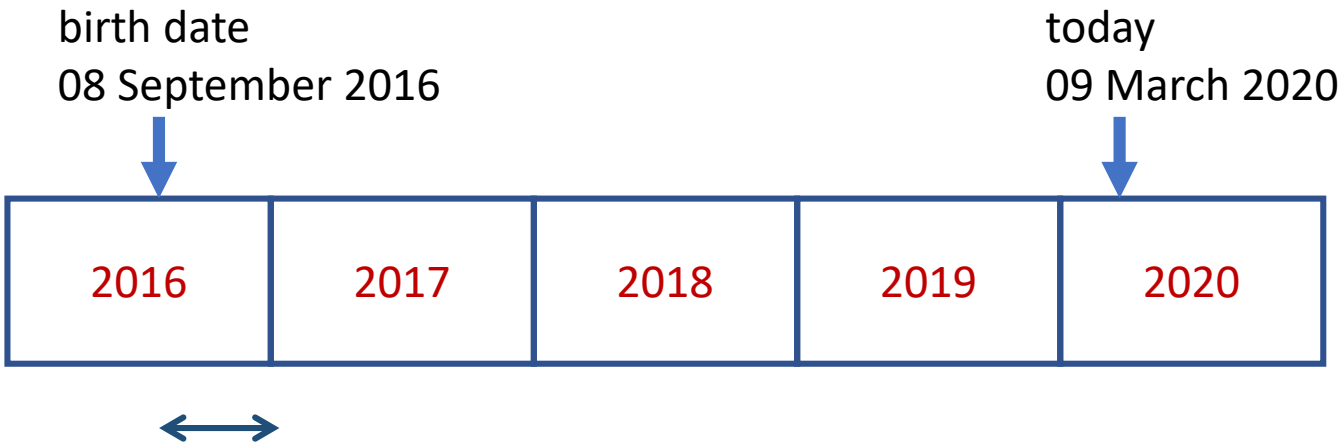
- Reviewing the DaysOfLife program - it's only an estimation
- Warm-up exercises. Swapping variable values, typecasting
- High-Level languages; compiled vs interpreted languages; reviewing IDLE and the Python Shell
- Libraries, e.g the math library

- Conditionals: if ... elif ... else
- Nested if statements
- Boolean comparisons and operators: not, and, or
- Specifying more complex conditions
- Two programming challenges - using if statements
  - A tennis scoring program
  - Perfecting the DaysOfLife program

```
# Get birthdate from user as 3 values: year, month & day
dobYear = int(input('Enter year of birth :'))
dobMonth = int(input('Enter month of birth (1 -12) :'))
dobDay = int(input('Enter day of birth: '))
# Establish today's date as 3 values: year, month & day
todayYear, todayMonth, todayDay = 2020, 3, 9
# Work out days in incomplete first year of life (daysFirstYear)
daysFirstYear =  (12-dobMonth) * 30 + (30-dobDay)
# Work out how many  days in whole years of life (wholeYears)
daysWholeYears = (todayYear - dobYear - 1) * 365  # calculate without leap years
# Work out days in incomplete current year of life (daysCurrentYear)
daysCurrentYear = todayDay + (todayMonth-1)*30 #assume roughly 30 days per month
# Estimate number of leap years (leapYears)
leapyears = (todayYear - dobYear)//4
# Rough answer = daysFirstYear + daysWholeYears + daysCurrentYear + leapyears
estDaysOfLife = daysFirstYear + daysWholeYears + daysCurrentYear + leapyears
print('You have been alive very roughly', estDaysOfLife, 'days')
```

Analyzing the days of life calculation

Take someone who was born on 8 September 2016 ...



daysFirstYear
* 30 - 8 days of Sept left
* 12 - 9 = 3 months left
@ 30 days each

daysWholeYears
* years = 2020 - 2016 - 1
* @365 days per year

daysCurrentYear
* whole months = 3 - 1 @
30days each
* 9 days of March

Warm-up exercise 1

- **Swapping two variable values**

Let's say in a program that you have two variables:

name1 *and* name2

What statements would you add to your program to swap them round so that name1 becomes whatever name2 is, and vice versa?

Warm-up exercise 1

- **Swapping two variable values**

Let's say in a program that you have two variables:

name1 *and* name2

What statements would you add to your program to swap them round so that name1 becomes whatever name2 is, and vice versa?

Warm up exercise 2

len(): a function which returns the length of a string provided

```
>>>len('hippopotamus')
        12
>>>len(4)
        error
```

```
>>> len(4)

Traceback (most recent call last):
  File "<pyshell#47>", line 1, in <module>
    len(4)
TypeError: object of type 'int' has no len()
```

Why does this error happen?

Write a program to take two integer numbers from the user, multiple them together and then report back on how many digits are in the answer.

HUMAN (PROGRAMMER)

natural language

---

high-level languages

Python

C, C++, C#, Java

these languages use **abstraction** to get closer to natural language and are easier for us to understand and debug, but less efficient and harder to optimize for the machine

assembly language

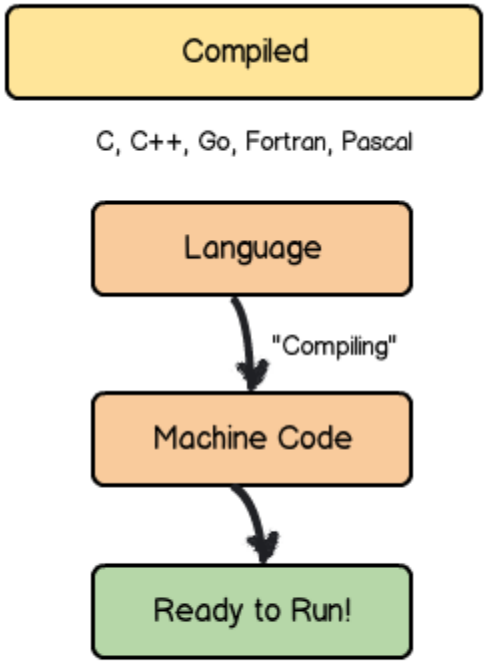
machine code

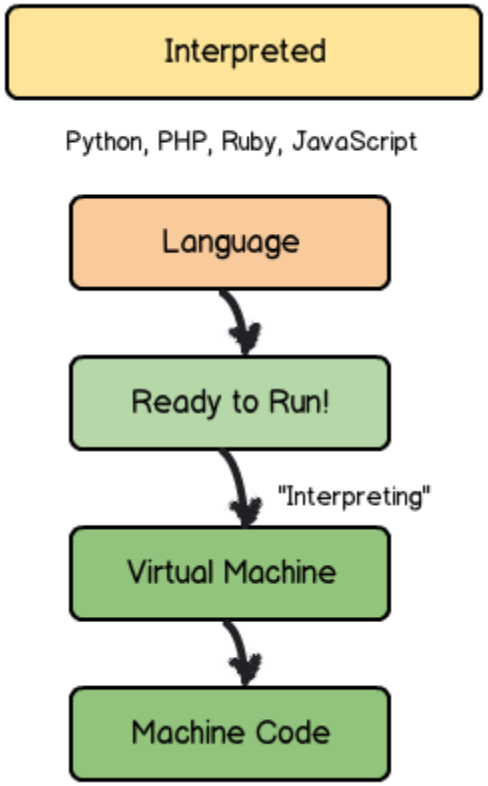
these languages are closer to the architecture of the machine, and are therefore more efficient and powerful to achieve optimisation

---

binary

COMPUTER


Compiled
C, C++, Go, Fortran, Pascal
Language
"Compiling"
Machine Code
Ready to Run!


Interpreted
Python, PHP, Ruby, JavaScript
Language
Ready to Run!
"Interpreting"
Virtual Machine
Machine Code

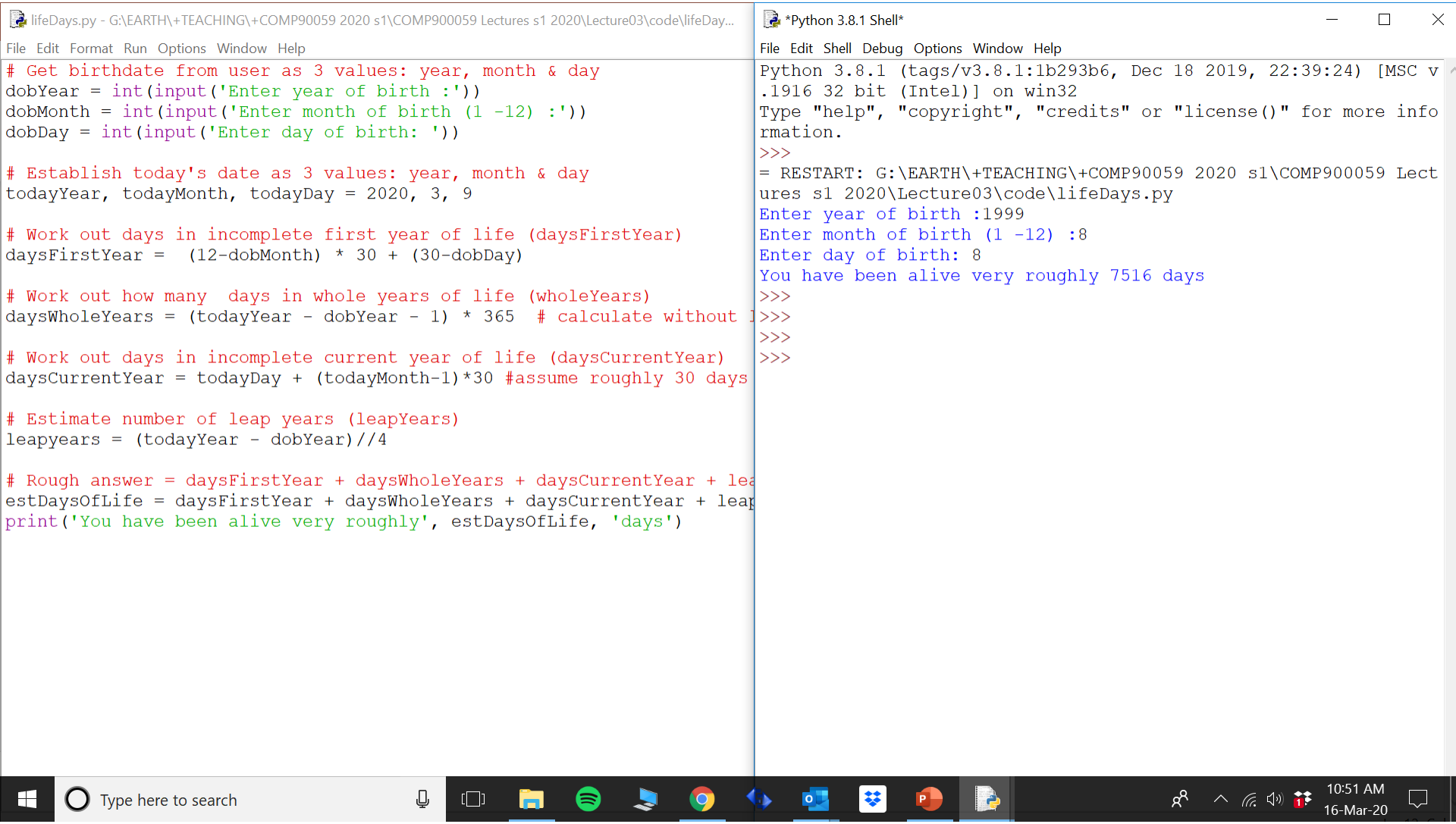
IDLE, scripts, & the Python shell ...

Python IDLE (Interactive DeveLopment Environment) - this is where you develop your programs ...

... you can write directly into the Python shell

... or save your code in a .py file, called a 'script' or 'module' and then run it

... double clicking on a .py file will run the script in a new window


lifeDays.py - G:\EARTH\+TEACHING\+COMP90059 2020 s1\COMP900059 Lectures s1 2020\Lecture03\code\lifeDay...
File Edit Format Run Options Window Help
# Get birthdate from user as 3 values: year, month & day
dobYear = int(input('Enter year of birth :'))
dobMonth = int(input('Enter month of birth (1 -12) :'))
dobDay = int(input('Enter day of birth: '))
# Establish today's date as 3 values: year, month & day
todayYear, todayMonth, todayDay = 2020, 3, 9
# Work out days in incomplete first year of life (daysFirstYear)
daysFirstYear = (12-dobMonth) * 30 + (30-dobDay)
# Work out how many days in whole years of life (wholeYears)
daysWholeYears = (todayYear - dobYear - 1) * 365 # calculate without
# Work out days in incomplete current year of life (daysCurrentYear)
daysCurrentYear = todayDay + (todayMonth-1)*30 #assume roughly 30 days
# Estimate number of leap years (leapYears)
leapyears = (todayYear - dobYear)//4
# Rough answer = daysFirstYear + daysWholeYears + daysCurrentYear + lea
estDaysOfLife = daysFirstYear + daysWholeYears + daysCurrentYear + leap
print('You have been alive very roughly', estDaysOfLife, 'days')
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v
.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more info
rmation.
>>>
= RESTART: G:\EARTH\+TEACHING\+COMP90059 2020 s1\COMP900059 Lect
ures s1 2020\Lecture03\code\lifeDays.py
Enter year of birth :1999
Enter month of birth (1 -12) :8
Enter day of birth: 8
You have been alive very roughly 7516 days
>>>
>>>
>>>
>>>
Type here to search
10:51 AM
16-Mar-20

lifeDays.py - G:\EARTH\+TEACHING\+COMP90059 2020 s1\COMP900059 Lectures s1 2020\Lecture03\code\lifeDay...

File Edit Format Run Options Window Help

```
# Get birthdate from user as 3 values: year, month & day
dobYear = int(input('Enter year of birth :'))
dobMonth = int(input('Enter month of birth (1 -12) :'))
dobDay = int(input('Enter day of birth: '))

# Establish today's date as 3 values: year, month & day
todayYear, todayMonth, todayDay = 2020, 3, 9

# Work out days in incomplete first year of life (daysFirstYear)
daysFirstYear =  (12-dobMonth) * 30 + (30-dobDay)

# Work out how many  days in whole years of life (wholeYears)
daysWholeYears = (todayYear - dobYear - 1) * 365  # calculate without l

# Work out days in incomplete current year of life (daysCurrentYear)
daysCurrentYear = todayDay + (todayMonth-1)*30 #assume roughly 30 days

# Estimate number of leap years (leapYears)
leapyears = (todayYear - dobYear)//4

# Rough answer = daysFirstYear + daysWholeYears + daysCurrentYear + lea
estDaysOfLife = daysFirstYear + daysWholeYears + daysCurrentYear + leap
print('You have been alive very roughly', estDaysOfLife, 'days')
print('daysWholeYears: ',daysWholeYears)
```

Python 3.8.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v
.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more info
rmation.
>>>
= RESTART: G:\EARTH\+TEACHING\+COMP90059 2020 s1\COMP900059 Lect
ures s1 2020\Lecture03\code\lifeDays.py
Enter year of birth :1999
Enter month of birth (1 -12) :8
Enter day of birth: 8
You have been alive very roughly 7516 days
>>>
>>>
>>>
>>>
= RESTART: G:\EARTH\+TEACHING\+COMP90059 2020 s1\COMP900059 Lect
ures s1 2020\Lecture03\code\lifeDays.py
Enter year of birth :
= RESTART: G:\EARTH\+TEACHING\+COMP90059 2020 s1\COMP900059 Lect
ures s1 2020\Lecture03\code\lifeDays.py
Enter year of birth :1999
Enter month of birth (1 -12) :8
Enter day of birth: 8
You have been alive very roughly 7516 days
daysWholeYears:  7300
>>>
>>> x = 4
>>> y = 5
>>> 5//4
1
>>> 5/4
1.25
>>> name = 'Johnson'
>>> name
'Johnson'
>>>
```

Libraries, modules and functions

- Module - the name for program code stored in a file
- Libraries - are special modules that contain an extra set of valuable functions that we can use
- For example, the Python math module/library contains many useful mathematical functions
- To use the math library in your program, you need to first tell Python ... add the following line to your code
  - import math
- Later in the program you can call any of its functions by adding 'math.' to the front
  - e.g., number = math.sqrt (200)

Using the math module in the python shell

```
>>> import math
>>> dir(math)
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec
__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil',
'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1'
, 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd',
'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamm
a', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radian
s', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
>>>
```

```
>>> help(round)
Help on built-in function round in module builtins:

round(number, ndigits=None)
    Round a number to a given precision in decimal digits.

    The return value is an integer if ndigits is omitted or None.  Otherwise
    the return value has the same type as the number.  ndigits may be negative.
```

- To use a resource from a module, you write the name of a module as a qualifier, followed by a dot (.) and the name of the resource
- – Example: math.pi

```
>>> print(math.pi * 2)
6.283185307179586
```

Exercise 3: using the maths module

- Write a program that takes as input the radius of a circle in cm and output to the screen the area of the circle in $cm^{2.}$
- area of circle = $\pi\ r^2$

```
import math
radius = float(input('Enter radius of circle in cms: '))
print('Area of circle', math.pi * radius ** 2,'cms squared')
```

```
>>> import math
>>> dir(math)
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec
__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil',
'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1'
, 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd',
'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamm
a', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radian
s', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
>>>
```

Conditionals

Conditionals

Conditionals are a core technique in all programming languages to specify what actions to take under different conditions

- *if a parking fine has been paid, send a thank you message*
- *if it has NOT been paid, send a reminder to pay*

if, else, elif ... are reserved words in Python to handle conditionals

Boolean logic - is the way we handle if conditions are true or not. Every conditional statement depends on boolean expressions to determine which action to take.

if statement

```
if<condition>:
        statement(s)
```

```
if country == 'Wales':
        print('Wales is my favourite country')

if parkingTicketPaid:
        print('Thank you!')

if temperature > 38:
        print('Warning, the temperature is over 38C')
```

Boolean: Comparisons and Expressions

```
if country == 'Wales':
        print('Wales is my favourite country')
if temperature > 38:
        print('Warning, the temperature is over 38C'
```

COMPARISON OPERATOR	MEANING
==	Equals
!=	Not equals
<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal

if ... else statement

```
if<condition>:
        statement(s)
else:
        statement(s)
```

```
if parkingTicketPaid:
        print('Thank you!')
else:
        print('Please pay your parking ticket.')

if temperature > 38:
        print('Warning, the temperature is over 38C.')
        print('You should cease working now.')
else:
        print('The temperature is below 38C.')
        print('You are fine to continue working.')
```

Exercises 4 & 5: finding the biggest number

Write a program that takes in 2 numbers (num1, num2)

and tells us which is larger number.

Write a program that takes in 3 numbers (num1, num2, num3)
and tells us which is the largest number.

if ...elif ... else statement

```
if<condition>:
        statement(s)
elif <condition)>:
        statement(s)
else:
        statement(s)
```

Note the syntax ...

: (the colon)

indentation

code blocks

```
if temperature > 38:
        print('Warning, the temperature is over 38C.')
        print('You should cease working now.')
elif temperature == 38
        print ('The temperature is now 38C.')
        print('A warning will be sent if it rises further.')
else:
        print('The temperature is below 38C.')
        print('You are fine to continue working.')
```

Exercise 6: A message to the investor

Write a program to take from the user information about an investment (the amount invested initially, the number of years invested, and the interest rate which is assumed to be fixed, and a target sum of funds that they would like to reach by the end of the investment). Send a message back to the user to say whether the final balance will reach the target or not, with information about the excess or shortfall, or if it will meet the target exactly.

Use the formula for the future value of an investment ...

future value = principal sum x $(1 + \text{interest rate})^{\text{years}}$

where interest rate is expressed as a fraction, e.g. 5% becomes 0.05

Operator precedence for highest to lowest

TYPE OF OPERATOR	OPERATOR SYMBOL
Exponentiation	`**`
Arithmetic negation	`-`
Multiplication, division, remainder	`*`, `/`, `%`
Addition, subtraction	`+`, `-`
Comparison	`==`, `!=`, `<`, `>`, `<=`, `>=`
Logical negation	`not`
Logical conjunction and disjunction	`and`, `or`
Assignment	`=`

Boolean Logical Operators

Conditionals can have more complex conditions that are built up using Boolean operators:

and tests if two conditions are both true

if years > 2 and principal < 1000:

or tests if two conditions are either both true or if one is true (an inclusive OR)

if years > 2 or principal < 1000:

not tests if the negation (the opposite) of a single condition is true

if not (years > 2):

Boolean Logical Operators

Boolean operators can be used in combination to create expressions - with the overall expression being either true or false:

```
if  not(x = 1 and y < 4) or z = 2:
```

Is the above expression true or false, when ...

x = 1, y = 2, z = 3

Precedence rules: not first, then and, then or
... but better to use parentheses to make it very clear!

Boolean Quiz

- #Let A = True and B = False. Evaluate the following:

```
a.  A != B
b.  A and B
c.  A or B
d.  not A
e.  (A == B) or (A and B)
f.  A and not B
g.  ‘pot’ in ‘hippopotamus’
```

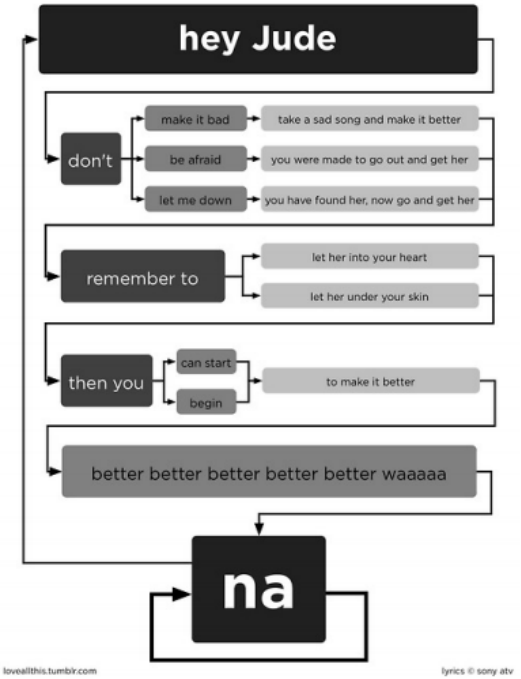
Exercise 7: Scoring a game of tennis. Part 1

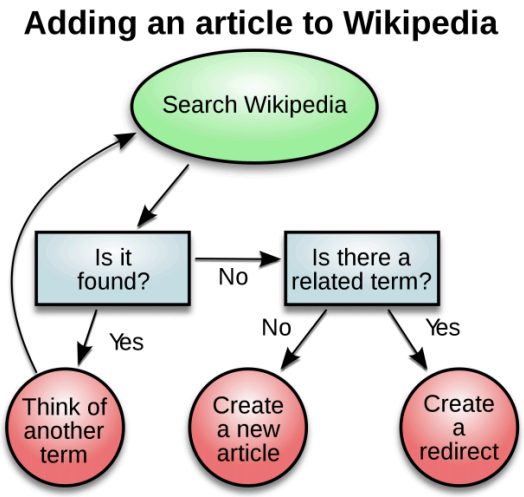
In a game of tennis, the score for player A (scoreA) and player B (scoreB) can be one of the following string values: *love, fifteen, thirty, forty, advantage*. Each player wins when they get to *forty* and they win the next point, unless they both have *forty* points.

Write a python statement that determines if player A has won the game based on three variables: scoreA and scoreB, and another variable called pointWinner which is set to 'A' or 'B' depending who has won the most recent point.

Program Design - solving the high level problem

Computational problem solving should start with a thorough analysis of the problem. Once the problem precisely understood then an ordered set of instructions are produced to solve the problem systematically.





repeat
search Wikipedia for the candidate article
if article found **then**
think of another term
else if article found for related term **then**
create a redirect
else
create a new article
end if
until article created **or** redirect created

Wikipedia Source: http://commons.wikimedia.org/wiki/File:Wikipedia_article-creation-2.svg

Hey Jude Source: https://laughingsquid.com/hey-jude-flow-chart/

Exercise 8: Leap Year

Write a program to take any year from the user, and say if it is a leap year or not.

"In the Gregorian calendar, a normal year consists of 365 days. Because the actual length of a sidereal year (the time required for the Earth to revolve once about the Sun) is actually 365.25635 days, a "leap year" of 366 days is used once every four years to eliminate the error caused by three normal (but short) years. Any year that is evenly divisible by 4 is a leap year: for example, 1988, 1992, and 1996 are leap years.

However, there is still a small error that must be accounted for. To eliminate this error, the Gregorian calendar stipulates that a year that is evenly divisible by 100 (for example, 1900) is a leap year only if it is also evenly divisible by 400.

For this reason, the following years are not leap years:

1700, 1800, 1900, 2100, 2200, 2300, 2500, 2600

This is because they are evenly divisible by 100 but not by 400.

The following years are leap years: 1600, 2000, 2400

This is because they are evenly divisible by both 100 and 400"

https://docs.microsoft.com/en-us/office/troubleshoot/excel/determine-a-leap-year

Exercise 9: Scoring a game of tennis. Part 2

Write a program that takes in the number of points that two tennis players have scored against each other, and gives the game score so far.

Player A	Player B	Score
0	0	Love All
1	0	Fifteen - Love
1	2	Fifteen - Thirty
2	2	Thirty All
2	3	Thirty - Forty
2	4	Player B wins
3	3	Deuce
3	4	Advantage Player B
4	4	Deuce
4	5	Advantage Player B
6	5	Advantage Player A
7	5	Player A wins
9	11	Player B wins

Exercise 10: Perfecting the DaysOfLife program

Make the DaysOfLife program code perfectly accurate, taking into account the exact number of days in each months, and taking into account leap years.

To do this use ..

- if, elif, else statements
- the leap year calculation we developed in this lecture

Tip: Look at each point in the current DaysOfLife code where an approximation is used. Think how you could precede that part of the code with a series of if ... elif ... else statements that reset the approximate value to the correct value each time the calculation is made.

This requires several lines of extra code to be added to specify the correct situation for each month, or for each year under consideration. My solution adds about 50 extra lines of code.

In later lectures we will learn how to solve this in a more elegant way!