

Getting Started With Python Programming: Part I

- Tutorial: creating computer programs
- Variables
- Getting information from the user
- Common mathematical operators

Reminder!

- These course notes are mandatory
 - Get them before class and go over them before attending
- (If all else fails then look through them afterwards – at the very least to see what concepts/topics you are responsible for knowing).
 - It's the **first** step you should complete if you've missed lecture and need to catch up.
 - (The second step is to get the in class notes of a classmate).
 - After going through these notes the third step is to ask us for help in filling in any conceptual gaps.

James Tam

Tips For Success: Programming Sections

- (The previous 4 tips are still applicable but there's some tips specific to programming):
 - **Take extensive notes:** everything in class not just what the instructor “writes down” but also what he/she “says in class”.
 - Some students may find when studying the lecture slides for the exam that they cannot understand concepts.
 - The extra “filling of the blanks” occurs during lecture so you need to annotate the slides with your own notes
 - After lectures have covered a particular concept/example
 - **If you have time try writing example programs on your own** (without looking at the online examples or notes) in order to create a program that fulfills the same task as the example program
 - (It's one thing to see the solution to a problem explained, your depth of understanding will be deeper if you have to re-create it from scratch yourself).
 - JT's note: you may find this unnecessary for the simple examples in this section but it will be beneficial to do this when more complex concepts are covered (e.g. nested loops onwards)

James Tam

Python

- This is the name of the programming language that will be used to illustrate different programming concepts this semester:
 - My examples will be written in Python
 - Your assignments will be written in Python
- Some advantages (from Python dot org)
 - Free
 - Powerful
 - Widely used (Google, NASA, Yahoo, Electronic Arts, some Linux operating system scripts etc.)
- Some starting python resources
 - Official website: <http://www.python.org>
 - An overview of the web site: <https://www.python.org/about/gettingstarted/>

James Tam

Python History

- Developed in the early 1990s by Guido van Rossum.
- Python was designed with a tradeoff in mind (from “*Python for everyone*” (Horstman and Necaise):
 - Pro: Python programmers could quickly *write programs* (and not be burdened with an overly difficult language)
 - Con: Python programs weren’t optimized to *run* as efficiently as programs written in some other languages.



“Gawky and proud of it.”

From:
<http://www.python.org/~guido/>

James Tam

Working From Home (Installing Python): Windows

- Getting Python (*get version 3.X and not version 2.X*)
 - <http://www.python.org/download/>

- Detailed information:
 - https://cspages.ucalgary.ca/~tam/resources/python/installing_accessing_python.pdf

James Tam

Online Help: Official Python Site

- *Basic explanation* of concepts (for beginners: along with examples to illustrate)
 - <http://docs.python.org/py3k/tutorial/index.html>
 - (Skip the notes on the interactive mode for now – it's where you don't save the program which leaves you nothing to submit for assignments).
 - For this course you need to create a python program in a file and then run the program defined in the file.

James Tam

Creating A Computer Program

1. A programmer writes the instructions of the program in high level (human can read and understand) language.

- Examples: C, C++, java, python

```
# Details later this term
list = [1,2,'a']
for element in list
  print(element)
```

2. The program must be created and saved using a text editor (e.g. Notepad, WordPad or the editor that comes with python IDLE).
 - Don't use a word processor.
3. The program is then translated into binary/machine language (the only form that the computer can understand).

```
# Details in 2nd year
10000001
10010100 10000100
10000001 01010100
```

James Tam

Location Of My Online Examples

- For this semester you can find them in D2L under: Content -> Lectures
- Then look under the appropriately named folder which is listed by date and topic.
- Alternatively you can find them by looking under the “main grid” of the course website (look for the ‘examples’ link):
 - [https://pages.cpsc.ucalgary.ca/~tamj/2024/231P/#Main_grid: course schedule for the lecture, lecture notes, assignment information](https://pages.cpsc.ucalgary.ca/~tamj/2024/231P/#Main_grid:_course_schedule_for_the_lecture,lecture_notes,assignment_information)

James Tam

The First Python Program

Name of the full example: 1small.py

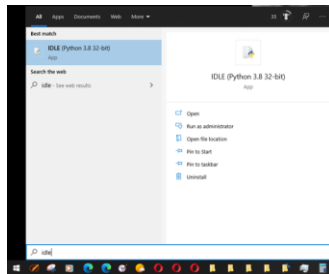
Filename: 1small.py

```
print ("hello")
```

James Tam

Creating/Running Programs: Windows

- **Step 0: Before writing your program start IDLE (python editor)**
 - The editor automatically comes installed with your python download so it's the one that we will officially support.
 - You can use other editors but keep in mind **you may be on your own if you have problems with that editor!**
 - Starting IDLE (Windows)



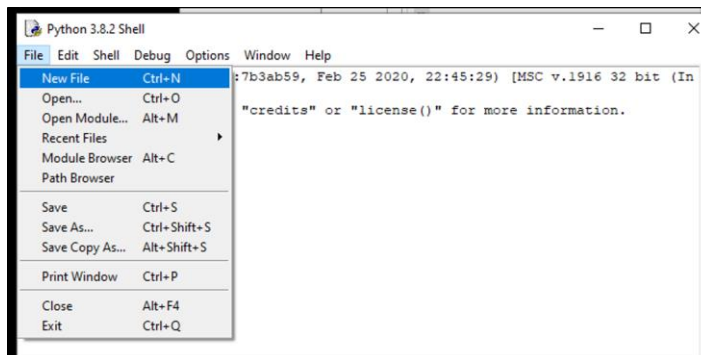
–One Apple resource for using IDLE:

<https://www.python.org/download/mac/tcltk/>

James Tam

Creating/Running Programs: Windows (2)

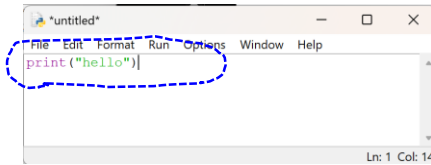
- **Step 1: Starting the python program editor from IDLE: File->New File**



James Tam

Creating/Running Programs: Windows (3)

- **Step 2:** Type your program into the editor window that appears (appears if you completed the previous step correctly):

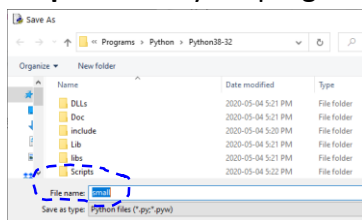


- (The specifics of what to type into the editor will form the bulk of teaching material for this semester – details are coming soon).
 - For now you can just type in what you see in the above image.

James Tam

Creating/Running Programs: Windows (4)

- **Step 3:** Name your program: File -> Save



- Using the IDLE editor you can simply save under the default file type.
 - (If you use another text editor such as WordPad then you will have to save it as a text file and make sure the filename ends with the right suffix “.py”).
 - Save the program in a easy to remember location (don't use the default location – it's the install location of python and not easy to remember for most).
- **Standards for naming programs**
 - Only use alphabetic or numeric characters in the save name
 - Don't include spaces or any other characters!

James Tam

Creating/Running Programs: Windows (5)

- **Step 4:** Running your program
 - Via the menu: Run->Run Module
 - Via key board shortcut: F5

James Tam

Opening Previously Created Programs

- Do this through the editor (IDLE):
- Menu options: File->Open
- **Do not try to 'click' on the file (Windows users)**
 - Windows: by default the operating system will try to run python programs (not allow you to edit it).

James Tam

Embedded Video (In Case You Can't See It Via The Link On The Last Screen).



James Tam

Requirements For Naming The File Containing Your Program

- As mentioned: no spaces!
 - Okay: `A1.py`, `gradeProgram.py`
 - Not Okay: `grade program.py`
- Just stick to using alphabetic characters, numbers may be used but the first character should only be alphabetic.
 - Okay: `A1Tam.py`, `A1Sept12.py`
 - Not Okay: `A1(1).py`, `9-25-2015.py`

James Tam

Section Summary: Writing A Small “Hello World” Program

- You should know exactly what is required to create/run a simple, executable Python program.
 - While you may not be able to create a new program from scratch at this point, you should be able to enter/run `small.py` yourself.
- You should also become familiar with requirements for naming the file which contains your python program.

James Tam

Variables

- Set aside a location in memory.
- Used to store information (temporary).
 - This location can store one ‘piece’ of information.
 - Putting another piece of information at an existing location *overwrites* previous information.
 - *At most* the information will be accessible as long as the program runs i.e., it’s temporary
- Some types of information which can be stored in variables include:
 - integer (whole number storage only) e.g. `age = 37`
 - floating point (fractional) e.g. `height = 68.5`
 - strings (character information - essentially any characters you can type and more) e.g. `fightingName = "TamJet"` (enclosed in double quotes – **do not use single quotes**)

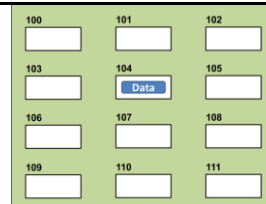


Image courtesy of James Tam

slide 22

James Tam

The Assignment Operator: =

- The assignment operator '=' used in writing computer programs does not have the same meaning as mathematics.
 - Don't mix them up!
- Example:
 - y = 3 (what is stored in 'y' at this point)
 - x = y (what is stored in 'x','y' at this point)
 - y = 6 (what is stored in 'x','y' at this point)
- What is the end result? How was this derived (what are the intermediate)
- **Name of the full example (shows output):** 2assignment.py

```
3
3 3
3 6
```

- Quick tip after using the assignment operator: to show what a variable currently contains put the name of the variable without quotes inside the round brackets for the print function e.g.,
 - num = 888
 - print(num)

James Tam

Displaying Output Using The Print() Function:

- This function takes zero or more arguments (inputs)
 - Multiple arguments are separated with commas (extra blank appears in the output) `print("hello", "there")` `hello there`
 - `print()` will display all the arguments followed by a blank line (the cursor is automatically moved to the next line).
 - Zero arguments just displays a blank line
- **Name of the full example:** 3outputExtras.py

```
print("hello", "there") hello there
```

```
print()
```

```
print("hello")
```

```
print("there")
```

James Tam

Side Note, Output: Python 3 Vs. Python 2

- Python 3: to be consistent all functions require brackets to enclose the arguments/function inputs
 - E.g. `print("Sup?")`
- Python 2 does not explicitly bracket all functions
 - E.g. `print "Sup?"`
- There are other differences so **make sure you are writing programs that follow the Python 3 syntax (rules)** and not Python 2 syntax.

James Tam

Arithmetic Operators

- Name of the full example: `6operators.py`

Operator	Description	Example
=	Assignment	<code>num = 7</code>
+	Addition	<code>num = 2 + 2</code>
-	Subtraction	<code>num = 6 - 4</code>
*	Multiplication	<code>num = 5 * 4</code>
/	Division (real number)	<code>num = 9 / 2</code> 4.5
//	Integer division	<code>num = 9 // 2</code> 4
%	Modulo (remainder)	<code>num = 9 % 2</code> 1
**	Exponent	<code>num = 9 ** 2</code> 81

James Tam

Order Of Operation

- First level of precedence: top to bottom
- Second level of precedence
 - If there are multiple operations that are on the same level then precedence goes from left to right.
 - **Name of the full example:** 7order . py

1 st	()	Brackets (inner before outer)
2 nd	**	Exponent
3 rd	*, /, //, %	Multiplication, division, modulo
4 th	+, -	Addition, subtraction
5 th	=	Assignment

Example

num = 3 * 2 ** 3

Vs.

num = (3 * 2) ** 3

James Tam

Order Of Operation And Style

- Even for languages where there are clear rules of precedence (e.g., Java, Python) it's good style to explicitly bracket your operations and use blank spaces as separators.


```
x = (a * b) + (c / d)
```
- It not only makes it easier to read complex formulas but also a good habit for languages where precedence is not always clear (e.g., C++, C).

James Tam

After This Section You Should Now Know

- How to create, translate and run Python programs.
- Variables:
 - What they are used for
 - How to access and change the value of a variable
 - How information is stored differently with different types of variables,
- Output:
 - How to display messages that are a constant string or the value stored in a memory location (variable or constant) onscreen with `print()`
- How/why use triple quoted output
- What are the Python operators for common mathematical operations
- How do the precedence rules/order of operation work in Python

James Tam