


Python development environment

Getting started with Python

- Install Python on your computer
- The latest release of Python version 3
- Download the Python software from:
 - <https://www.python.org/download>
- Select the appropriate installer for your operating system (Windows, Linux/Unix, Mac OS X)
- Find and click on the IDLE shortcut 

The Python shell

- Whether in Command Window or IDLE, Python is waiting for us to “say” something

`>>>` (this is the Python prompt)

- In this mode, Python will instantaneously INTERPRET our statements and return an “answer”

Open IDLE

What are the expected results for:

```
>>> 8 + 9.3
```

```
>>> 8 - 9.3
```

```
>>> 8 * 9.3
```

```
>>> 8 / 9.3
```

```
>>> 8 ** 9.3
```

Arithmetic expressions, data types and functions

Data Types / Python

- Numeric values
 - Integer
 - Values without a decimal point
 - Python data type **int**
 - Floating point
 - Values with a decimal point
 - Python data type **float**

Numerical Operations / Symbols

Operation	Symbol
Addition	+
Subtraction	-
Multiplication	*
Division (returns float)	/
Division Quotient (returns int)	//
Division Remainder (returns int)	%
Exponentiation	**

Operator precedence

*** / // %**

+ -

()

*** / // %**

+ -

Parentheses have precedence
over the other operators

What does $2 * 3 ** 2$ evaluate to?

Result Data Type

DT	Op	DT	Result DT
int	+ - *	int	int
int	/	int	float
int	//	int	int
int	%	int	int
float	%	float	float
int	**	int	int
float	**	int	float

Practice

- For each of the following expressions predict the result and it's data type; then execute the expression in IDLE to determine if you prediction is correct.
 - $4 + 2$
 - $4.0 + 2$
 - $6 * 7 - 9/3$
 - $6 * 7 - 3$
 - $2.5 / (1 + 4)$
 - $6.2 + 8 * 3 - 10/5$
 - $(6.2 + 8) * (3 - 10/5)$

Built-in functions

<code>abs(x)</code>	Returns the absolute value of a number
<code>float(x)</code>	Returns a floating point number constructed from a number
<code>format(value [, format_spec])</code>	Convert a value to a “formatted” representation, as controlled by <code>format_spec</code>
<code>int(x)</code>	Returns an integer value constructed from a number
<code>min (arg1, arg2,...)</code>	Returns the smallest of two or more arguments
<code>max(arg1, arg2,...)</code>	Returns the largest of two or more arguments
<code>round(number[, ndigits])</code>	Returns the floating point value number rounded to <code>ndigits</code> after the decimal point

<https://docs.python.org/3/library/functions.html>

Python Standard Library

<https://docs.python.org/3/library/>

Numeric and Mathematical Modules

- math
- fractions
- random

```
>>> import math
```

```
>>> math.pi
```

```
>>> 3.141592653589793
```

Variables

Variables

- Suppose we performed a complex calculation in Python and we wanted to SAVE the result for future use
- In order to do that we need to have a “container” in memory where we can store our piece of data
- Such “containers” are called **variables**

Variables: assignment statements

Formula for calculating compound interest

$$A = P \left(1 + \frac{r}{n}\right)^{nt}$$

A is the value after t periods

P is the principal amount (initial investment)

r is the annual nominal interest rate (not reflecting the compounding)

n is the number of times the interest is compounded per year

t is the number of years the money is invested

Given: n = 12 months, t = 10 years, p = 5000, r = 1/100

Go to IDLE

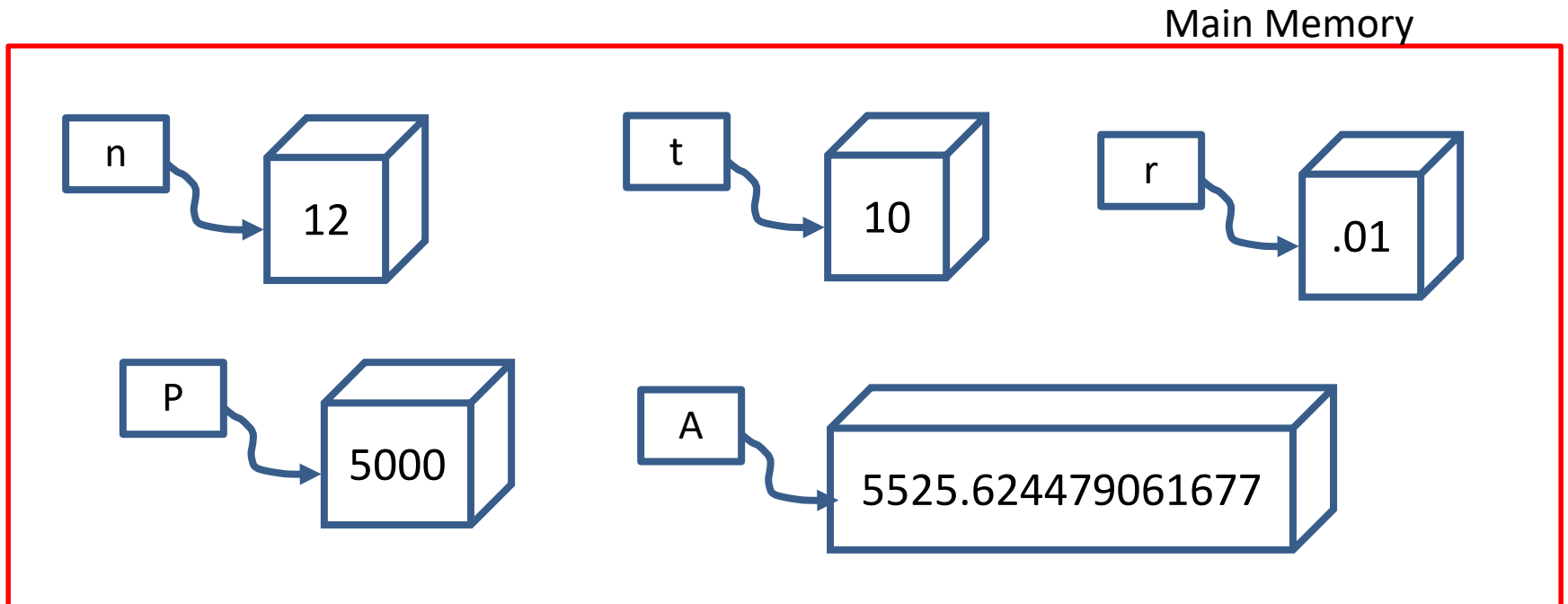
```
>>> A = 5000 * (1 + (1/100)/12) ** (12*10)
>>> print(A)
5525.624479061677
>>> #what does A represent
>>> print('The investment balance after 10 years is ', A)
The investment balance after 10 years is 5525.624479061677
>>> #want balance to 2 decimal places
>>> balance = round(A,2)
>>> print('The investment balance after 10 years is ', balance)
The investment balance after 10 years is 5525.62
>>>
```


Go to IDLE

```
>>> P = 5000
>>> r = 1/100
>>> n = 12
>>> t = 10
>>> A = P * (1 + r/n) ** (n*t)
>>> print(A)
5525.624479061677
>>> print('The investment balance after 10 years is ', A)
The investment balance after 10 years is 5525.624479061677
>>> balance = round(A,2)
>>> print('The investment balance after 10 years is ', balance)
The investment balance after 10 years is 5525.62
>>>
```

Variables: assignment statements

- We have done the following in memory for each variable:
 - An object (container) of type **int** was created
 - We put a label “n” on the object
 - The number 12 was stored as this object



What is the type for label A?

Legal Python variable names

- A variable name, can be made up of
 - Letters
 - Digits
 - The underscore character _
- It cannot begin with a digit
- It can be as long as you want
- Variable names are case sensitive
 - MyStuff and mystuff are different variables
 - Knowing this is important
 - Relying on this is confusing and error prone

Reserved words

- Besides variable names, that we make up, there are other words that make up programs
- Sometimes we are using another programmer's code, so we use the identifiers that she chose (such as print)
- Often we use special identifiers call reserved words that already have a predefined meaning in the language
- A reserved word cannot be used in any other way, for example as a variable name

Python reserved words

and	elif	if	not
as	else	import	or
assert	except	in	pass
break	false	is	raise
class	finally	int	return
continue	for	lambda	true
def	from	none	try
del	global	nonlocal	while
			with
			yield

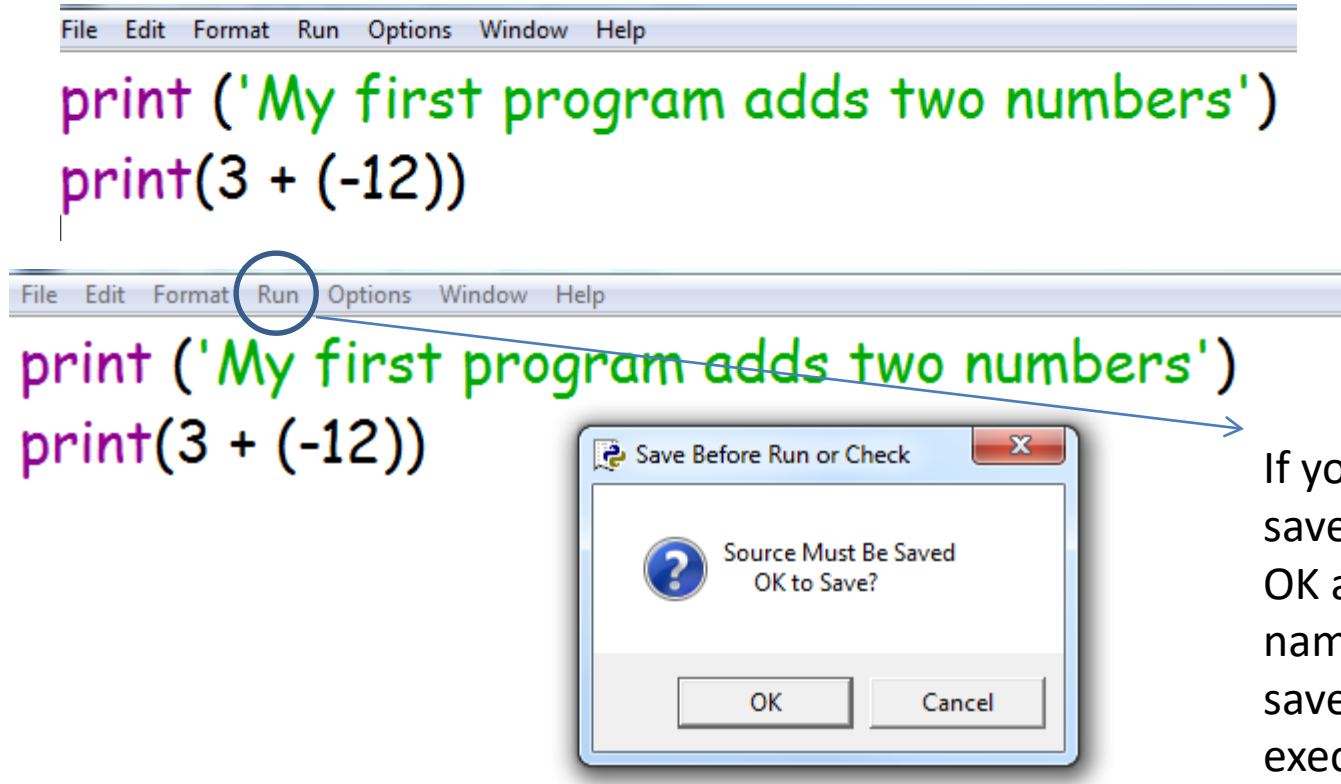
Multi-word variable names

- There are two conventions for names with multiple words:
 - Using underscore as delimiter
 - shoe_size
 - Using “camelCase” capitalizing the initial of the second (third,...) word
 - shoeSize
 - You can choose one style and stick with it
 - Be consistent within each program

Python program

Go to IDLE

Click File → New File



If you have not already saved the program, click OK and give this file a name. When you click save the program will execute

IDLE Shell

- The output is returned to the IDLE Shell

```
===== RESTART: C:/myPythonPrograms/FirstProgram.py :  
My first program adds two numbers  
-9
```

Variables: assignment statements

Formula for calculating compound interest

$$A = P \left(1 + \frac{r}{n}\right)^{nt}$$

A is the value after t periods

P is the principal amount (initial investment)

r is the annual nominal interest rate (not reflecting the compounding)

n is the number of times the interest is compounded per year

t is the number of years the money is invested

Given: n = 12 months, t = 10 years, p = 5000, r = 1/100

CompoundInterest Program

File Edit Format Run Options Window Help

```
# Written by: Dolores Kalayta
```

```
# Assign values to variables
```

```
P = 5000
```

```
r = 1/100
```

```
n = 12 # number of months
```

```
t = 10 # number of years
```

```
A = P * (1 + r/n) ** (n * t) # compound interest expression
```

```
balance = round(A,2)
```

```
print('The investment balance after 10 years is ', balance)
```

```
print('The interest amount is ', round((A - P), 2))
```

The investment balance after 10 years is 5525.62

The interest amount is 525.62

Practice Problem

- You scored $x/100$, $y/100$, $z/100$ on homework
 - What is your total homework percentage?

Create algorithm

- We want the program to use 3 variables and calculate the average percentage
 - Step 1: Assign a value to score_1
 - Step 2: Assign a value to score_2
 - Step 3: Assign a value to score_3
 - Step 4: Calculate Average by summing the scores and dividing by total possible points
 - Step 5: Multiply Average by 100 to get percentage
 - Step 6: Print percentage

Boolean expressions and operators

Boolean Expressions

- How does Python interpret an expression like the following?

```
>>> 5 <= 18
```

- Python can evaluate Boolean expressions, i.e. expression whose value is TRUE or FALSE
- Python interprets `5 <= 18` as a statement whose truth is being questioned

```
>>> 5 <= 18
```

```
True
```

bool

- Boolean data types, i.e. True or False, are called **bool** in Python
- Boolean expressions often contain COMPARISON operators

Operator	Meaning
<	Less than
>	Greater than
==	Equal to (TWO equal signs)
!=	Not equal to
<=	Less than or equal to
>=	Greater than or equal to

Boolean operators

- **and, or, not**
- You have seen these in discrete math classes
- **and, or** are binary operator
 - expression1 **and** expression2
 - expression1 **or** expression2
- **not** is a unary operator
 - **not** expression

Boolean operators

- expression1 **and** expression2
 - True if both expressions are true
 - False if any of the two expressions is false
- expression1 **or** expression2
 - True if any of the two expressions are false
 - False if both expression are false
- **not** expression
 - True if the expression is false
 - False if the expression is true

Operator Precedence

Arithmetic then comparison then boolean (N-A-O)

- ******
- ***** **/** **//** **%**
- **+** **-**
- **<**, **>**, **==**, **<=**, **>=**, **!=**
- **not**
- **and**
- **or**

Go to IDLE

- $2 < 3$ and $3 < 4$
- $4 == 5$ and $3 < 4$
- False and True
- True and True
- $\text{not}(3 < 4)$
- $\text{not}(\text{True})$
- $\text{not}(\text{False})$
- $4 + 1 == 5$ or $4 - 1 < 4$
- $4 + 1 == 5$ and $4 - 1 < 4$

Strings

Strings

- A string **value** is represented as a sequence of characters enclosed within quotes

```
>>> 'Hello World'
'Hello World'
```

- A string value can be assigned to a variable.

```
>>> s = 'rock'
>>> t = 'climbing'
```

- String values can be manipulated using string operators and functions

```
>>> s + t
'rockclimbing'
```

```
>>> s + ' ' + t
'rock climbing'
```

Explicit conversion of int to string

- Only string data can be concatenated
- To change an int data type to a string use str()
- Example:
- ```
>>> age = 16
>>> print('Adam is ' + age + ' years of age')
TypeError: must be str, not int
>>> print ('Adam is ' + str(age) + ' years of age')
Adam is 16 years of age
```

# First interactive Python program



# Variables: assignment statements

Formula for calculating compound interest

$$A = P \left(1 + \frac{r}{n}\right)^{nt}$$

A is the value after t periods

P is the principal amount (initial investment)

r is the annual nominal interest rate ( not reflecting the compounding)

n is the number of times the interest is compounded per year

t is the number of years the money is invested

Given: n = 12 months, t = 10 years, p = 5000, r = 1/100

# CompoundInterest Program

File Edit Format Run Options Window Help

```
Written by: Dolores Kalayta
```

```
Assign values to variables
```

```
P = 5000
```

```
r = 1/100
```

```
n = 12 # number of months
```

```
t = 10 # number of years
```

```
A = P * (1 + r/n) ** (n * t) # compound interest expression
```

```
balance = round(A,2)
```

```
print('The investment balance after 10 years is ', balance)
```

```
print ('The interest amount is ', round((A - P), 2))
```

The investment balance after 10 years is 5525.62

The interest amount is 525.62

# Problem

- Calculate the new balance and the interest earned based on the values specified by the user
- Need to get
  - Initial investment,  $P$
  - Annual interest rate,  $r$  (expressed as float)
  - Number of times the interest is compounded per year,  $n$
  - The number of years the money is invested,  $t$
- Return to user
  - New balance
  - Amount of interest earned

# input() function / eval() function

- To have the user input data:
  - `input('Enter name: ')` returns a string value
  - To save the data input, assign it to a variable
    - `firstName = input('Enter name: ')`
  - `x = input('Enter number: ')` returns a string
    - User types 4 – this is saved as '4' – cannot do a calculation on text (string) data
    - To convert to a number use the `eval()` function
      - `num = eval(x)`

# Practice Problem

- You scored  $x/100$ ,  $y/100$ ,  $z/100$  on homework
  - What is your total homework percentage?

Modify the previous program to get the input from the user.

- Calculate the area of a circle; let the user specify the radius of the circle.