Introduction to Python

—

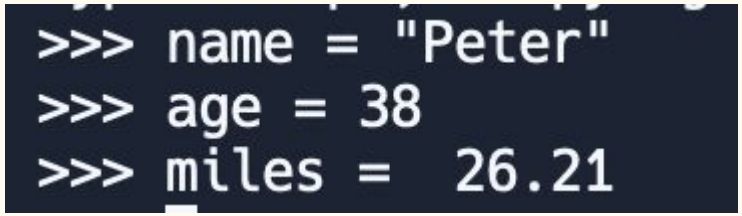
Variables

What is a variable?

A variable is a name that refers to a value. This means that when you create a variable you reserve some space in memory to store your value.

Python assign a value to a variable using the equal sign (=).

Here are some examples:



STRING

INTEGER

FLOATING POINT

Working with variables

We can use a print statement to display the value of a variable:

```
>>> print(name)
Peter
>>> print(age)
38
>>> print(miles)
26.21
```

The type of a variable is the type of the value it refers to.

Python uses the method type() to return the class type of the argument (object) passed as parameter

```
>>> type(name)
<class 'str'>
>>> type(age)
<class 'int'>
>>> type(miles)
<class 'float'>
```

Variable names

- Programmers generally choose names for their variables that are meaningful and document what the variable is used for.
- Variable names can be arbitrarily long.
- They can contain both letters and numbers, but they cannot start with a number.
- It is legal to use uppercase letters.
- Begin variable names with a lowercase letter.
- The underscore character (_) can appear in a name. It is often used in names with multiple words, such as my_name or favorite_color.
- Variable names can start with an underscore character, but we generally avoid doing this unless we are writing library code for others to use.

Reserved words

Remember special words have special meaning to Python.

Should **NOT** be used as a name for a variable.

and	as	assert	class	continue	def
del	elif	else	except	finally	for
from	global	if	import	in	is
lambda	nonlocal	not	or	pass	raise
return	try	while	with	yield	

Statements

A statement is a unit of code that the Python interpreter can execute. We have seen two kinds of statements: print being an expression statement and assignment.

A script usually contains a sequence of statements. If there is more than one statement, the results appear one at a time as the statements execute.

For example, the script

```
print(1)
x = 2
print(x)
```

produces the output

```
1
2
```

Operators and operands

Let's write in your script the code that prints our traditional Hello World! by typing it inside print() function.

Operators are special symbols that represent computations like addition and multiplication. The values the operator is applied to are called ***operands***.

+	addition
-	subtraction
*	multiplication
/	division
**	exponentiation

Expressions

An expression is a combination of values, variables, and operators.

A value all by itself is considered an expression, and so is a variable.

Examples (assuming that the variable x has been assigned a value):

```
15
x
x + 5
```

If you type the expression 1 +1 in the Python console, it will display 2 as output the interpreter evaluates it and displays the result.

However, an expression all by itself doesn't do anything in a script. So, if you type 1 + 1 in your script you will not get 2

Order of operations

For expression with more than one operator Python follows mathematical convention (PEMDAS)

Example:

2 * (3-1) = 4

1+1)**(5-2) = 8

What is the result for the following expressions?

INDIVIDUAL ACTIVITY

1. WORK ON “01 LAB - VARIABLES”
2. COMPLETE YOUR DAILY LOG PROGRAMMING

QUESTIONS?