alt-text

Intermediate Python

Allen Sanders

asanders@gamuttechnologysvcs.com

Before we begin, be sure you have Python 3 installed!

1. Install Python 3 if you have not already
    - go to https://www.python.org/downloads/
2. Install Visual Studio Code if not already installed
    - go to https://code.visualstudio.com/Download
3. Install Python extension in VS Code

Data Types, Operators, and Variables

Primitives and expressions

- Python provides a collection of primitive types
- Base types like integers, floats, strings, and bools
- Using operators, can also build out expressions to combine data in interesting ways

In [1]:

```
print(199)       # int
print(1.99)      # float
print('199')     # str
print(False)     # bool
```

```
199
1.99
199
False
```

In [2]:

```
print(3 + 8 * 2)
print((3 + 8) * 2)
```

```
19
22
```

Arithmetic Operators

Operation	Description
a + b	Addition
a - b	Subtraction
a * b	Multiplication
a / b	Division
a // b	Truncating Division
a ** b	Power (a raised to power of b)
a % b	Modulo (remainder)
-a	Unary minus
+a	Unary plus

In [3]:

```
a = 11
b = 3
c = -4
d = 11.8
e = 2.99
print(a / b)        # produces floating point number
print(a // b)       # floor division - truncates
print(d / e)
print(d // e)
print(a ** b)
print(a % b)
print(-c)
print(+c)
```

```
3.6666666666666665
3
```

```
3.9464882943143813
3.0
1331
2
4
-4
```

Common mathematic functions

Function	Description
abs(x)	Absolute value
divmod(x, y)	Returns (x // y, x % y)
pow(x, y)	Same as (x ** y)
round(x)	Round (uses "banker's rounding")

In [4]:

```
x = 11
y = 3
z = -18
a = 0.5
b = 1.5
c = 2.5
d = 3.5
print(abs(z))
print(divmod(x, y))
print(pow(x, y))
print(round(a))
print(round(b))
print(round(c))     # banker's rounding - round to nearest even multiple
print(round(d))
```

```
18
(3, 2)
1331
0
2
2
4
```

Shortcut operators

- Allows `h = h + 1` to be shortened to `h += 1`
- Supported for many of the operators discussed previously
- Python does not have an increment (++) or decrement (--) operator

Dynamic typing, no declarations

In [5]:
```
x:int = 3.9
print(x)
x = "Python"    # more likely to see this version
print(x)
```

```
3.9
Python
```

...but strongly typed

In [6]:
```
x = 'hello'
y = x + 1
y
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_22349/297019856.py in <module>
      1 x = 'hello'
----> 2 y = x + 1
      3 y

TypeError: can only concatenate str (not "int") to str
```

In [ ]:
```
def func(arg):
    return arg + 1

print(func(2))
print(func('hi'))
```

```
3
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_21281/3567252731.py in <module>
      3
      4 print(func(2))
----> 5 print(func('hi'))

/tmp/ipykernel_21281/3567252731.py in func(arg)
      1 def func(arg):
----> 2     return arg + 1
      3 
      4 print(func(2))
      5 print(func('hi'))

TypeError: can only concatenate str (not "int") to str
```

Console input and output

- To print output to the console, you can use the `print` command
- To accept input from the consle, you can use the `input` command
- An *f-string* can be used to format output

  See https://zetcode.com/python/fstring/#:~:text=Python%20f-string%20is%20the%20newest%20Python%20syntax%20to,prefix%20and%20use%20%7B%7D%20brackets%20to%20evaluate%20values.

In [ ]:

```
tax_rate = 0.075
quantity = int(input('How many are you purchasing? '))
cost = float(input('What is the unit cost? '))
total = quantity * cost
print(f'Total (without tax): ${total:,.2f}')
print(f'Total (with tax): ${total * (1 + tax_rate):,.2f}')
```

```
Total (without tax): $35.88
Total (with tax): $38.57
```

Text strings

- To define a string literal, can enclose in single, double, or triple quotes
- Same type of quote used to start the string must be used to terminate it
- Strings using single and double quotes must be limited to single logical line
- Triple-quoted strings allows text to span multiple lines

In [ ]:

```
name = 'Allen Sanders'
```

```
fav_color = "red"
fav_food = '''ice cream'''
fav_movie = """The Godfather"""
multi_line = '''This is line one

This is line two

This is line three'''

print('Name:', name)
print('Favorite Color:', fav_color)
print('Favorite Food:', fav_food)
print(multi_line)
```

```
Name: Allen Sanders
Favorite Color: red
Favorite Food: ice cream
This is line one

This is line two

This is line three
```

String operations

- String stored as sequence of Unicode characters
- Can be concatenated with `+`
- Individual characters can be accessed using 0-based integer index
- Multiple methods available for working with strings

Method	Description
s.startswith(val[, start[, end]])	Checks whether string s starts with "val"
s.endswith(val[, start[, end]])	Checks whether string s ends with "val"
s.find(sub[, start[, end]])	Finds 1st occurrence of "sub" in string s or returns -1 if not found
s.replace(old, new[, maxreplace])	Replaces substring "old" with "new"
s.split([sep[, maxsplit]])	Splits string s using separator "sep"
s.strip([chars])	Removes leading/trailing spaces of "chars" value from string

Method	Description
s.lower()	Converts string s to lowercase
s.upper()	Converts string s to uppercase

In [ ]:

```
s = 'Hello World'
print(s.startswith('Hello'))
print(s.endswith('world'))
print(s.lower().endswith('world'))
print(s.find('llo Wor'))
print(s.split())
```

```
True
False
True
2
['Hello', 'World']
```

Exercise One

- Create a Python program for processing user profile data inputs
- Prompt the user for input of the following data values:
  - First name
  - Last name
  - Age
  - Number of years of experience in current role
  - Job title
- Print the provided data to the screen in an organized format (your choice)

Exercise Two

- Create a Python program for processing an order
- Prompt the user for input of the following data values:
  - Part number
  - Quantity
  - Unit cost

        - Discount
- Using the provided inputs, calculate subtotal, total including tax, and final total after discount
- Print the formatted order detail to the screen