



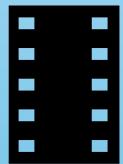
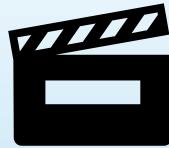
Paper 184-2025

Fun With SAS® and Emoji

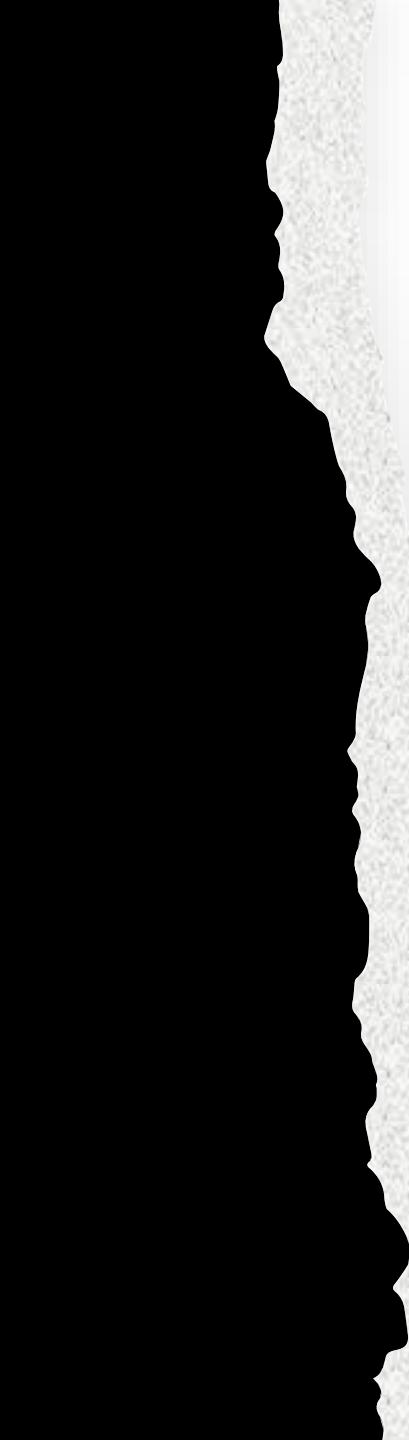
What Might a Rebus-Influenced Programming
Language Look Like?

SHOOT SCHEDULE

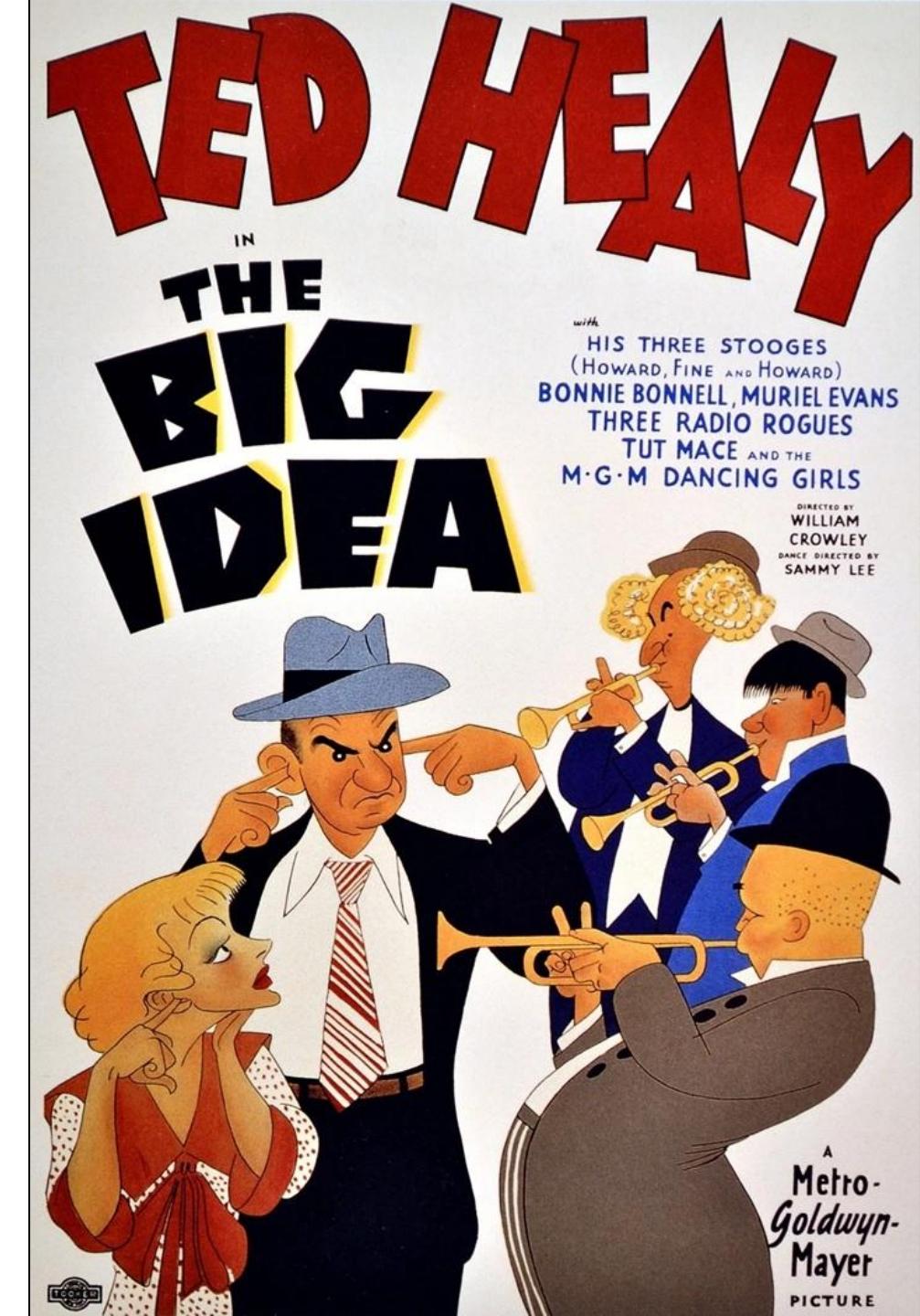
SCENE	DESCRIPTION
1	THE BIG IDEA
2	YOU OUGHT TO BE IN PICTURES
3	WORDS AND PICTURES
4	A SIMPLE PLAN
5	IT'S COMPLICATED
6	WHO YA GONNA CALL? SAS K FUNCTIONS!
7	WHO YA GONNA CALL II? PYTHON REGEX FUNCTIONS!
8	THAT'S ALL FOLKS!



Would You Use a
“Running Semicolon”
Character?



RUN;



Would You Use a "Running Semicolon" Character?

Posted 02-20-2022 04:06 PM (3248 views)



"I cringe at the notion..."

"Bad idea IMO..."

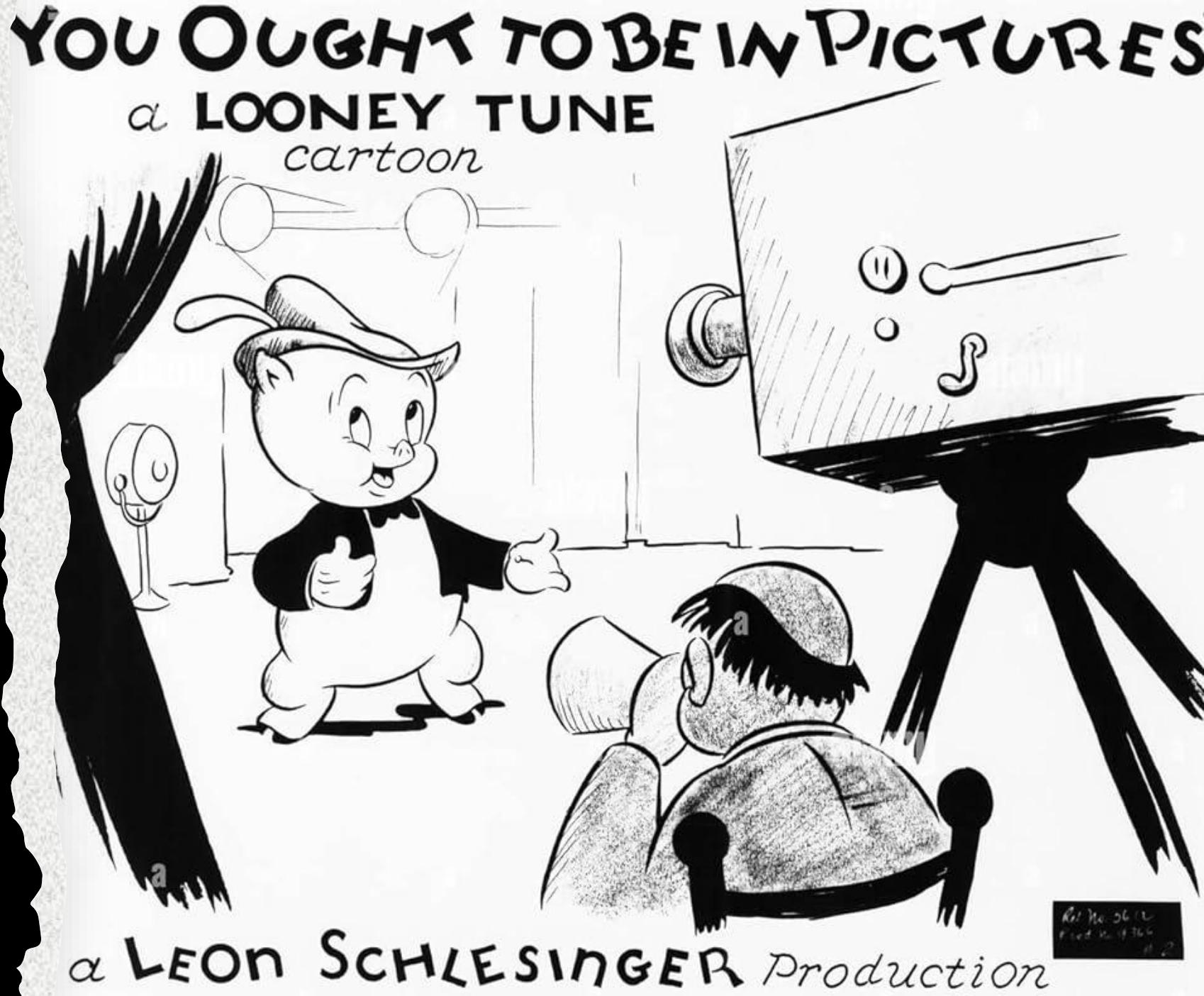
"Programming languages shouldn't have decoration: save that for places like Twitter..."

"Running Semicolon"

As HBO's Silicon Valley amusingly showed us, programmers can get into heated arguments over the use of [Tabs vs. Spaces](#).

In the SAS world, debates can arise over [whether every DATA step and PROC should be terminated with a RUN or QUIT statement](#) - large SAS programs may include 100+ "RUN;" statements - many technically 'unnecessary' (depending on who you ask!) - making some SAS programmers happy while irritating others. So, a question for lovers and haters of the RUN-even-if-unnecessary rule: If a "running semicolon" or "RUN character" was available, would you be okay with the use of that?

Code With Pictures? How? Why?



Highlights

Jimmy Goes to the Library

By Pat Kite

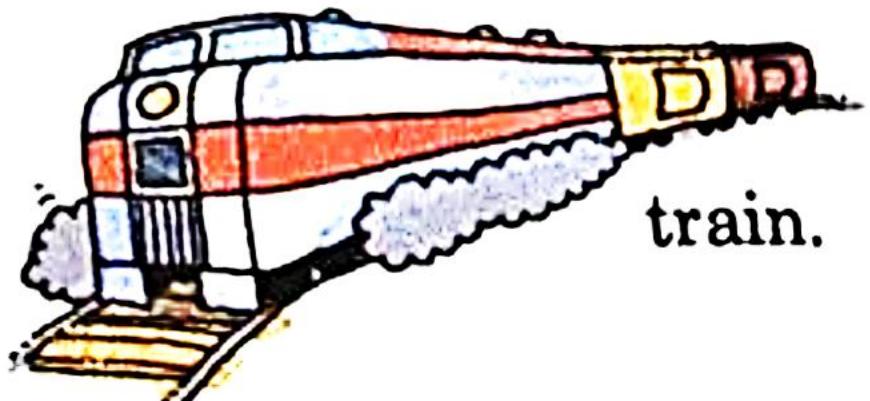


"Let's go to the library,"



They took a

"How will I be able to find the book I want?"



train.

Mother said.



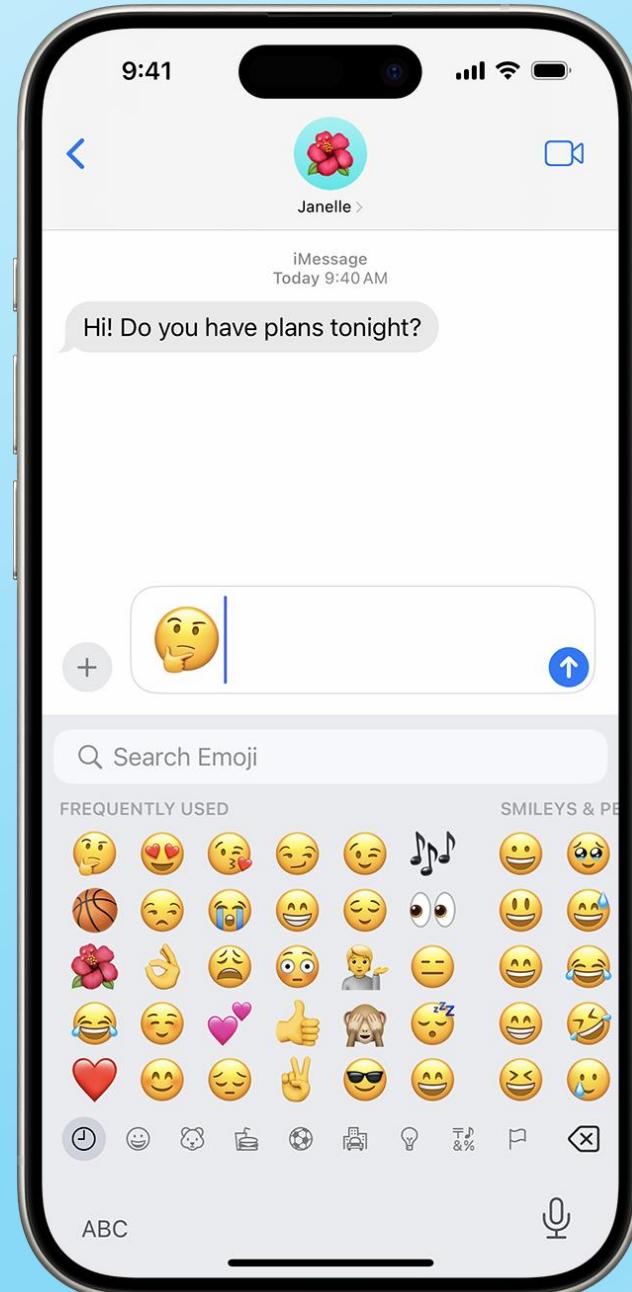
Jimmy asked.

Mother said.

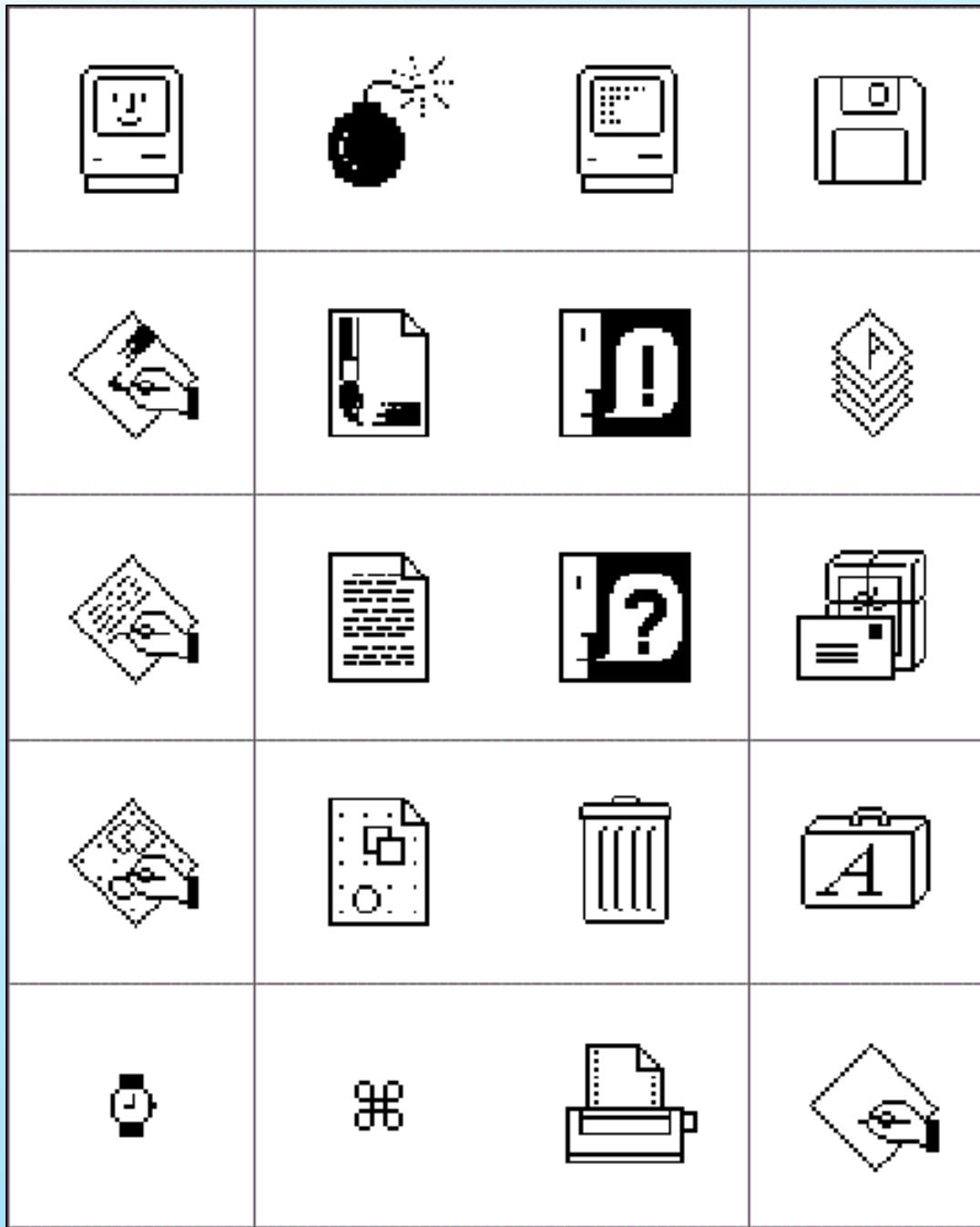




I  NY®



Early Mac Icons & Emoji by Susan Kare



Characters Generated by Cairo Type Font

.	1	2	3	4	5	6	7	8	9	0	-	=	Back Space
Tab	🌴	👁️	🍴	🍾	🍷	🌙	⭐	⊗	✖️	➡️	⬅️	↙	↖
Caps Lock	🅰️	🆂	Ԁ	🅁	🅶	🅃	🅄	🅅	🅆	🅾️	🅿️	🅾️	🅿️
Shift	;z	×	c	v	b	n	m	,	:	/	;	Shift	
	dog	+/-	frog	owl	balloons	babies	smiley	bars	bars	key	bell	gun	key
Opt	grid									Enter	Opt		
~	!	϶	*	\$	϶	&	*	()	-	+	Back Space	
Tab	🦆	🍦	☀️	leaf	apple	house	tree	star	rose	bag	↑	↑	↑
Caps Lock	Q	H	E	R	T	V	U	I	O	P	{	}	!`
Shift	z	x	c	u	b	n	m	<	>	?	;	Shift	
	rose	flower	grapes	ancient symbol	apple	strawberry	watch	glasses	key	key	;		
Opt	grid									Enter	Opt		

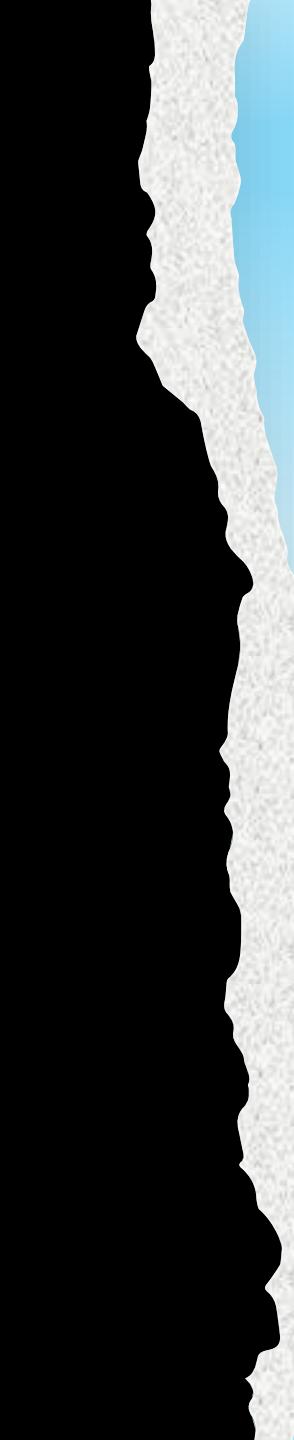
To Insert These Characters in Your Text
Select "Cairo" Font, Size 18 Point



life←{↑1 ωV.∧3 4=+/,-1 0
1o.⊖-1 0 1o.Φ⊂ω}

APL – A PROGRAMMING
LANGUAGE (Early 1960s)

So, What Might SAS Emoji Coding Look Like?

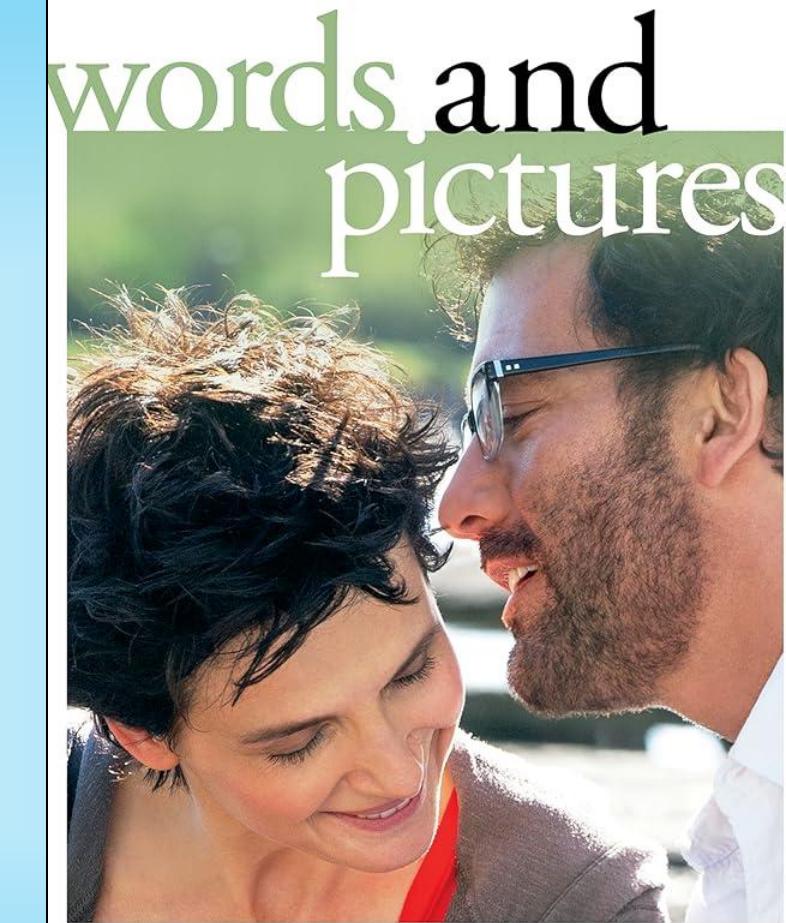


academy award® nominee

clive owen

academy award® winner

juliette binoche



is a man worth more than his words,
a woman worth more than her pictures?



DIE PRODUCTIONS IN ASSOCIATION WITH LASCAUX FILMS PRESENTS A FRED SCHEPISI FILM. CLIVE OWEN JULIETTE BINOCHE "WORDS AND PICTURES" BRUCE DAVIDSON NAVIO NAGAR
ANNEMAN CASTING DEBORAH AQUILA CSA TRISHA WOOD CSA ADRIANA MAUREEN WEBB COMPOSER PAUL GRABOWSKY DIRECTOR OF PHOTOGRAPHY TISH MONAGHAN EDITOR PETER HONESS A.C.E. PRODUCED BY PATRIZIA VON BRAND
IAN BAKER EXECUTIVE PRODUCER BOB GASS JUDY BURCH GASS JOSEPH COHEN RICHARD TOUSSAINT WADE BARKER DERRICK EVERIS DESIGNER NANCY RAE STONE PRODUCED BY GERALD DIPEGO FRED SCHEPISI
Latitude DOOLBY DOLBY ATMOS CURTIS BURCH MUSIC GERALD DIPEGO WRITTEN BY FRED SCHEPISI
WordsAndPicturesTheMovie.com #WordsAndPicturesMovie



if g=h then i=j; else k=substr(L, 1, 3);



g=h



i=j;



k=



(L, 1, 3);

SAS Code (Text)

```
data test;  
set;  
if a=b then c=d; else e=f;  
if g=h then i=j; else k=substr(l, 1, 3);  
dt=date(); tm=time();  
proc print; run;
```

SAS Code (Text+Emoji)

```
 test;  
set;  
 a=b  c=d;  e=f;  
 g=h  i=j;  k=  (l, 1, 3);  
dt=  (); tm=  ();  
proc  ; 
```



```
select make from cars group by 1 order by 1 desc;
```



make



cars $\sum 1$



1



;

SQL Code (Text)

```
proc sql nopr.int;  
create table car_summary as  
select make, model,  
max(dt) as maxdate format=date10.,  
min(tm) as mintime format=time8.  
from sashelp.cars t1  
join dt_tm t2 on 1=1  
group by make, model  
order by make, model desc;
```

SQL Code (Text+Emoji)

```
proc sql noprint;  
─ car_summary as  
🛒 make, model,  
max(dt) as max 📅 format=📅 10.,  
min(tm) as min ⏳ format=⌚ 8.  
⬅ sashelp.cars t1  
🔗 dt_tm t2 on 1=1  
Σ make, model  
⇅ make, model ⏴ ;
```

SAS EG Code Snippet Abbreviations

The image illustrates a workflow for creating and using a SAS code snippet in SAS Enterprise Guide (EG).

- Creating a Snippet:** A screenshot of the SAS EG interface shows the "Program" menu open. The "New snippet" option under the "Project" submenu is highlighted with a blue circle containing the number 1.
- Snippet Definition:** A "New Snippet" dialog box is displayed. It contains fields for "Abbreviation" (set to "export") and "Text to insert" (containing the PROC EXPORT code). A checkbox for "SAS code editing mode" is checked. The "OK" button is visible at the bottom right, with a blue circle containing the number 2.
- Using the Snippet:** In the SAS EG code editor, the abbreviation "exp" is typed into the code area. A dropdown menu appears, listing several code snippets, with "FOOTNOTE" being selected, indicated by a blue circle containing the number 3.
- Resulting Code:** The final screenshot shows the expanded PROC EXPORT code in the code editor, with the insertion point positioned after the identifier "FOOTNOTE". The "OK" button is visible at the bottom right, with a blue circle containing the number 4.

copilotunicodesas.sas *wuss2025example.sas test_pgm_in.sas adventofcodeday1.sas dad.sas eclipse2024.sas *emmys2024.sas

CODE LOG OUTPUT DATA

Line #

```
22 ods graphics / width=16in height=9in noborder outputfmt=svg; * SVG format to ma
23 proc sgpanel data=emmys noautolegend des="2024 Emmy Awards"; * Draw charts for r
24 panelby award / headerattrs=(weight=bold color=black) novarname nowall noheaderborder nobo
25 hbarparm category=seqx response=probability / nooutline fillattrs=(color=cXA8D8FF); * Bar c
26 text x=minX y=seqx text=nominee / position=right strip contributeoffsets=none textattrs=(si
27 text x=maxX y=seqx text=odds / position=left strip contributeoffsets=none textattrs=(size=9
28 rowaxis display=(nolabel novalues noticks noline); * Suppress most axis labels
29 colaxis display=(nolabel noticks) values=(0 .1 .2 .3) valuesdisplay=('0%' '10%' '20%' '30%
30 run;
31
```

/home/ted.conway/emmys2024.sas

Line 17, Column 41

UTF-8

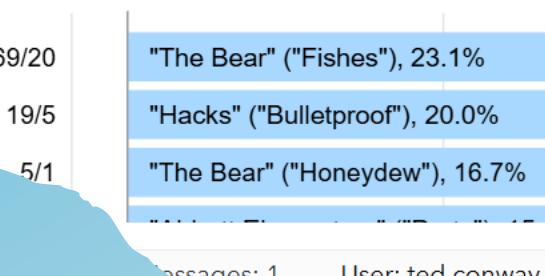
RESULTS

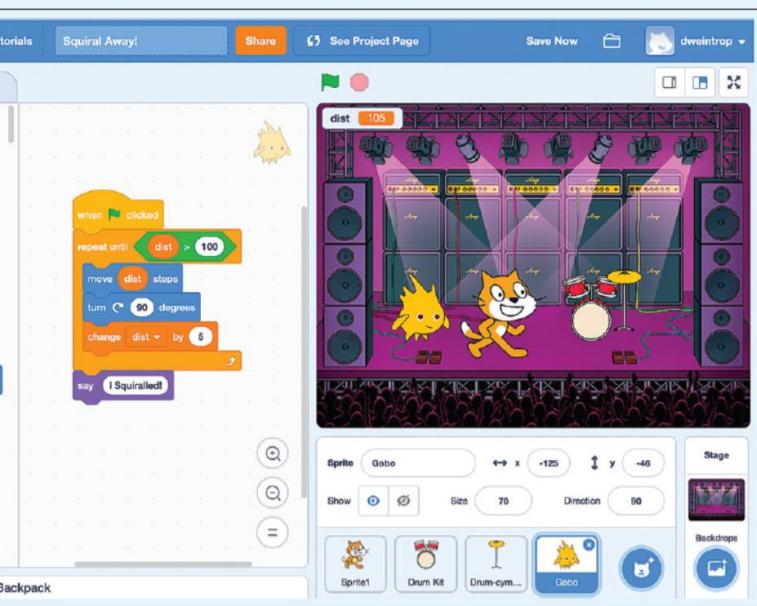


Contents

SAS Syntax Highlighting

Best Comedy Actress





(a)



(b)

```

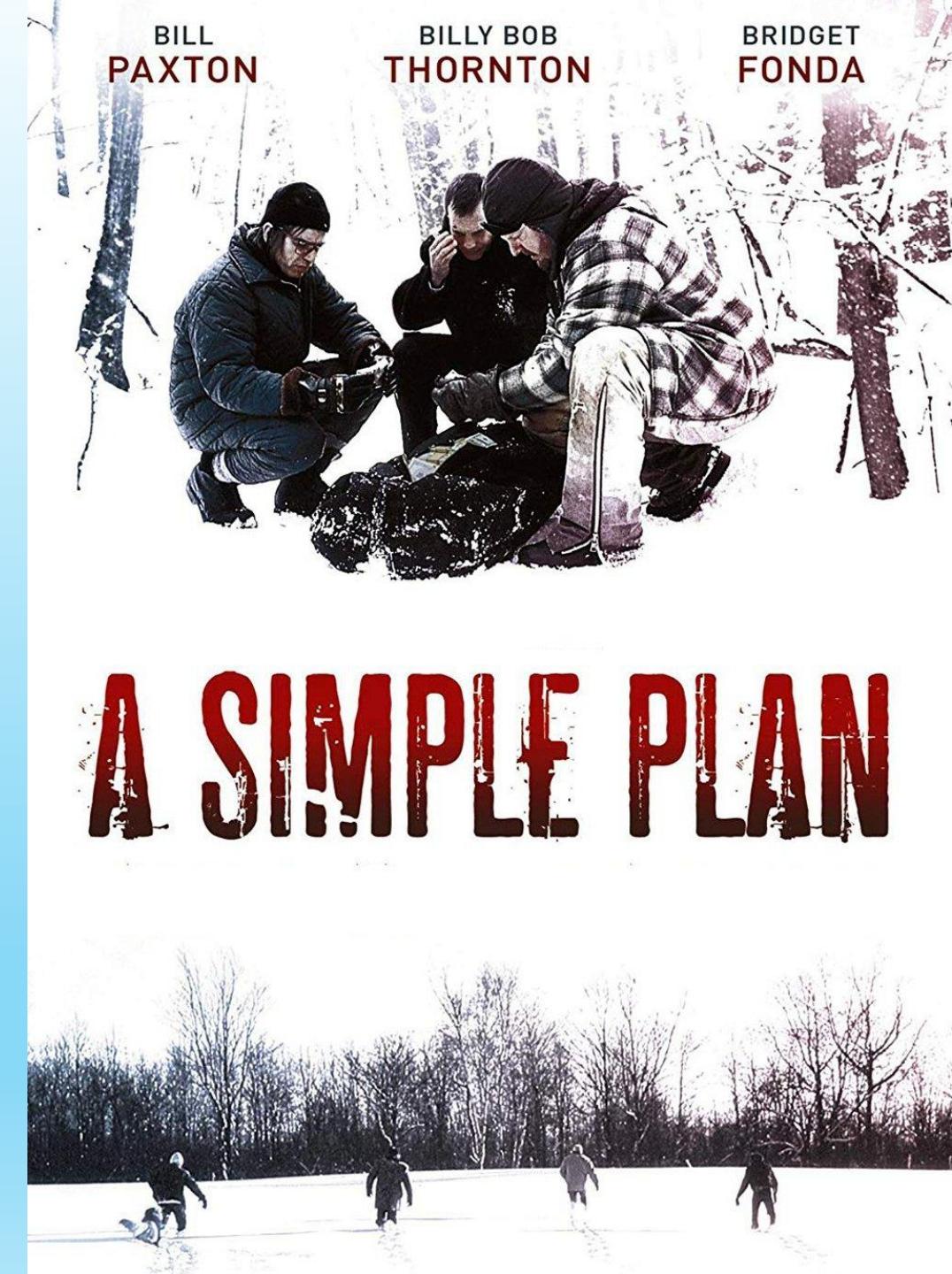
answer_tests.py U display_tests.py U
src > quizzes > tests > display_tests.py > DisplayQuizViewTests > test_quiz_redirects

4 from django.test import TestCase
5 from django.urls import reverse
6
7
8 class DisplayQuizViewTests(TestCase):
9     def test_quiz_404(self):
10         url = reverse("quizzes:display_quiz", args=(12,))
11         response = self.client.get(url)
12         self.assertEqual(response.status_code, 404)
13
14     def test_quiz_redirects(self):
15         quiz, question, _ = create_quiz()
16         url = reverse("quizzes:display_quiz", args=(quiz.pk,))
17         response = self.client.get(url)
18         self.assertRedirects(response, reverse("quizzes:display_question", args=(quiz.pk, question.pk)))
19
20
21 class DisplayQuestionViewTests(TestCase):
22     def test_quiz_404(self):
23         url = reverse("quizzes:display_question", args=(12, 1))
24         response = self.client.get(url)
25         self.assertEqual(response.status_code, 404)
26
27     def test_question_404(self):
28         quiz, question, _ = create_quiz()
29         url = reverse("quizzes:display_question", args=(quiz.pk, question.pk + 100))
30         response = self.client.get(url)

```

Block-Based vs. Text Programming

A Simple Emoji-to-Text Preprocessor

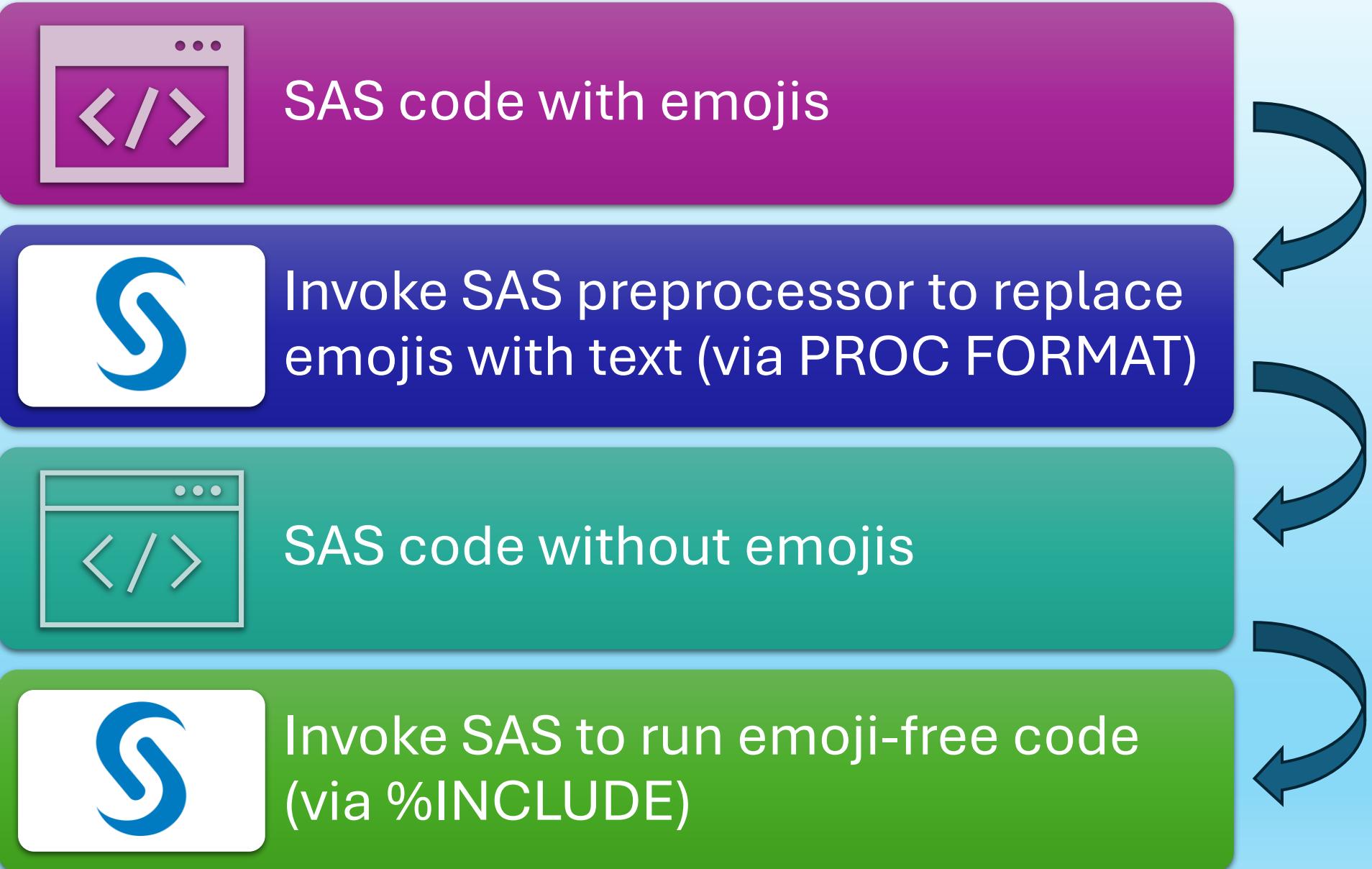


Nice idea, but...

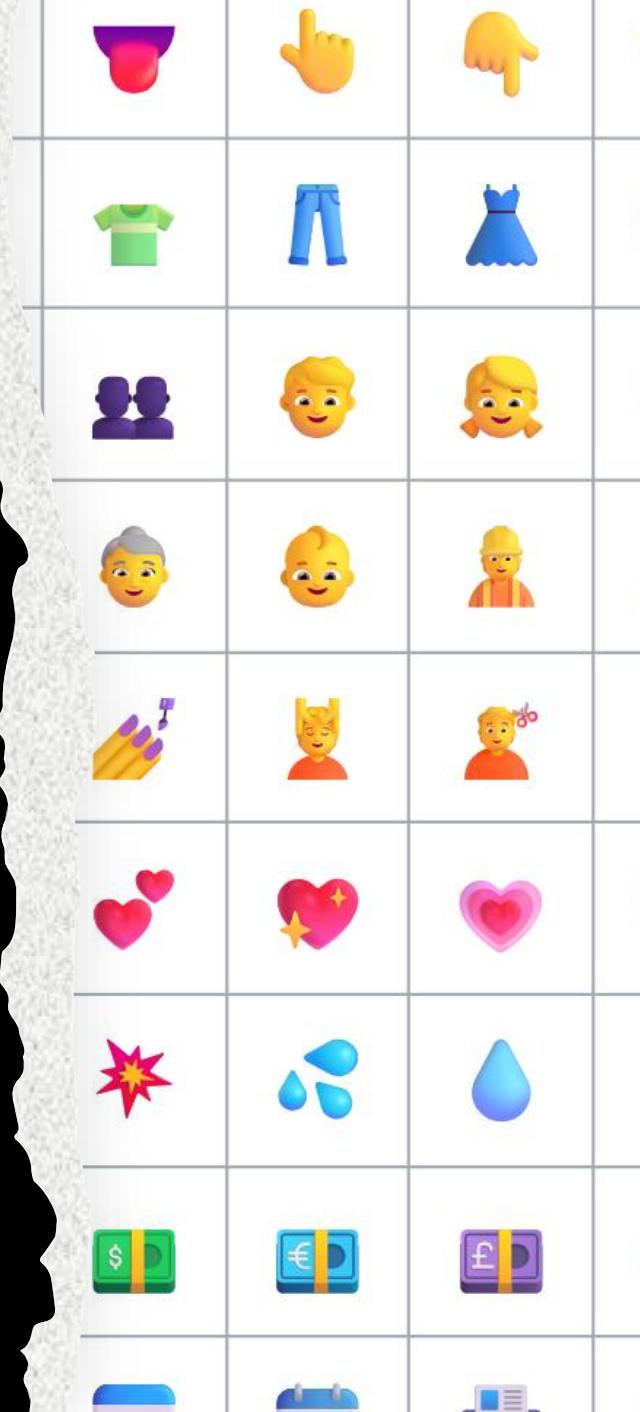
- ✖ SAS macro variable names **must begin with a letter or an underscore**
- ✖ Macro names are **subject to standard SAS naming conventions**

```
%macro  / stmt;  
substr  
%mend;  
  
data _null_;  
x =  ("abcdef", 2, 3);  
run;
```

Plan B – A SAS Preprocessor



Emojis are complicated!

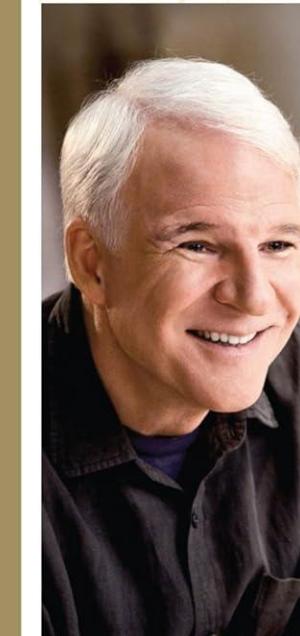


From the Writer/Director of
SOMETHING'S GOTTA GIVE & THE HOLIDAY

MERYL
STREEP

STEVE
MARTIN

ALEC
BALDWIN



Written and Directed by Nancy Meyers

it's
Complicated



UNIVERSAL

Emoji Encoding

- In the Unicode and UTF-8 world of emoji, **one character may now occupy 1-4 bytes**
- Complex graphemes built from **multiple code points, joiners & modifiers** require even more!

Character	Description	Code Points (Sequence)	UTF-8 Encoding (Hex)	Bytes
A	Latin Capital A	U+0041	41	1
🤔	Thinking Face	U+1F914	F0 9F A4 94	4
✓	White Heavy Check Mark	U+2705	E2 9C 85	3
✗	Cross Mark Button	U+274E	E2 9D 8E	3
🏃	Person Running	U+1F3C3	F0 9F 8F 83	4
🏃	Man Running Facing Right (ZWJ seq)	U+1F3C3 [🏃] U+200D U+2642 [♂] U+FE0F U+200D U+27A1 [→] U+FE0F	F0 9F 8F 83 E2 80 8D E2 99 82 EF B8 8F E2 80 8D E2 9E A1 EF B8 8F	22

SAS K Functions

- Traditional SAS string functions are single-byte oriented
 - ✓ LENGTH(' 😊 😨 🤝 ')=12, Not 3!
- K functions operate on characters, not bytes (K is short for Kanji)
- KSUBTR, KLENGTH, KINDEX, etc.
 - ✓ KSUBSTR(' 😊 😨 🤝 ',2,1)=' 😨 '
 - ✓ KLENGTH(' 😊 😨 🤝 ')=3
- K functions handle single Unicode code points (🥺) but not more complex grapheme clusters (🏃)

WHO YA GONNA CALL?



BILL MURRAY DAN AYKROYD
SIGOURNEY WEAVER

GH~~O~~STBUSTERS

COLUMBIA PICTURES PRESENTS AN IVAN REITMAN FILM A BLACK RHINO/BERNIE BRILLSTEIN PRODUCTION
BILL MURRAY DAN AYKROYD SIGOURNEY WEAVER "GHOSTBUSTERS"
ALSO STARRING HAROLD RAMIS RICK MORANIS MUSIC BY ELMER BERNSTEIN GHOSTBUSTERS PERFORMED BY RAY PARKER, JR.
PRODUCTION DESIGN BY JOHN DE CUIR DIRECTOR OF PHOTOGRAPHY LASZLO KOVACS, ASC VISUAL EFFECTS BY RICHARD EDLUND, ASC
EXECUTIVE PRODUCER BERNIE BRILLSTEIN WRITTEN BY DAN AYKROYD AND HAROLD RAMIS PRODUCED AND DIRECTED BY IVAN REITMAN
PCP PARENTAL GUIDANCE SUGGESTED. SOME MATERIAL MAY NOT BE SUITABLE FOR CHILDREN.

ORIGINAL SOUNDTRACK ALBUM ON ARISTA RECORDS

SONY

make.believe

COLUMBIA PICTURES

SAS PREPROCESSOR (CODE)

```
proc format; value $emojif  
  '💾'='data' '❓'='if' '✅'='then' '✖'='else' '🏃'='run';  
  
data _null_;  
length txt varchar(1024);  
infile "/home/ted.conway/test_pgm_in.sas"  
      lrecl=32767 recfm=n truncover length=l;  
file "/home/ted.conway/test_pgm_out.sas";  
input pgm $varying32767. l;  
do c=1 to klength(pgm);  
  txt=put(ksubstr(pgm,c,1),$emojif.);  
  txt_len=length(txt);  
  put txt $varying. txt_len@;  
end;  
run;  
%include "/home/ted.conway/test_pgm_out.sas";  
run;
```

PROC FORMAT
MAPPING

READ SAS CODE
WITH EMOJIS

REPLACE
EMOJIS W/TEXT

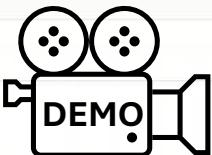
RUN EMOJI-FREE
SAS CODE

test_pgm_in.sas WussSasDemo.sas

CODE LOG

1 _null_;
2 a=1 b=1; b=0;

Cut code



/home/ted.conway/test_pgm_in.sas

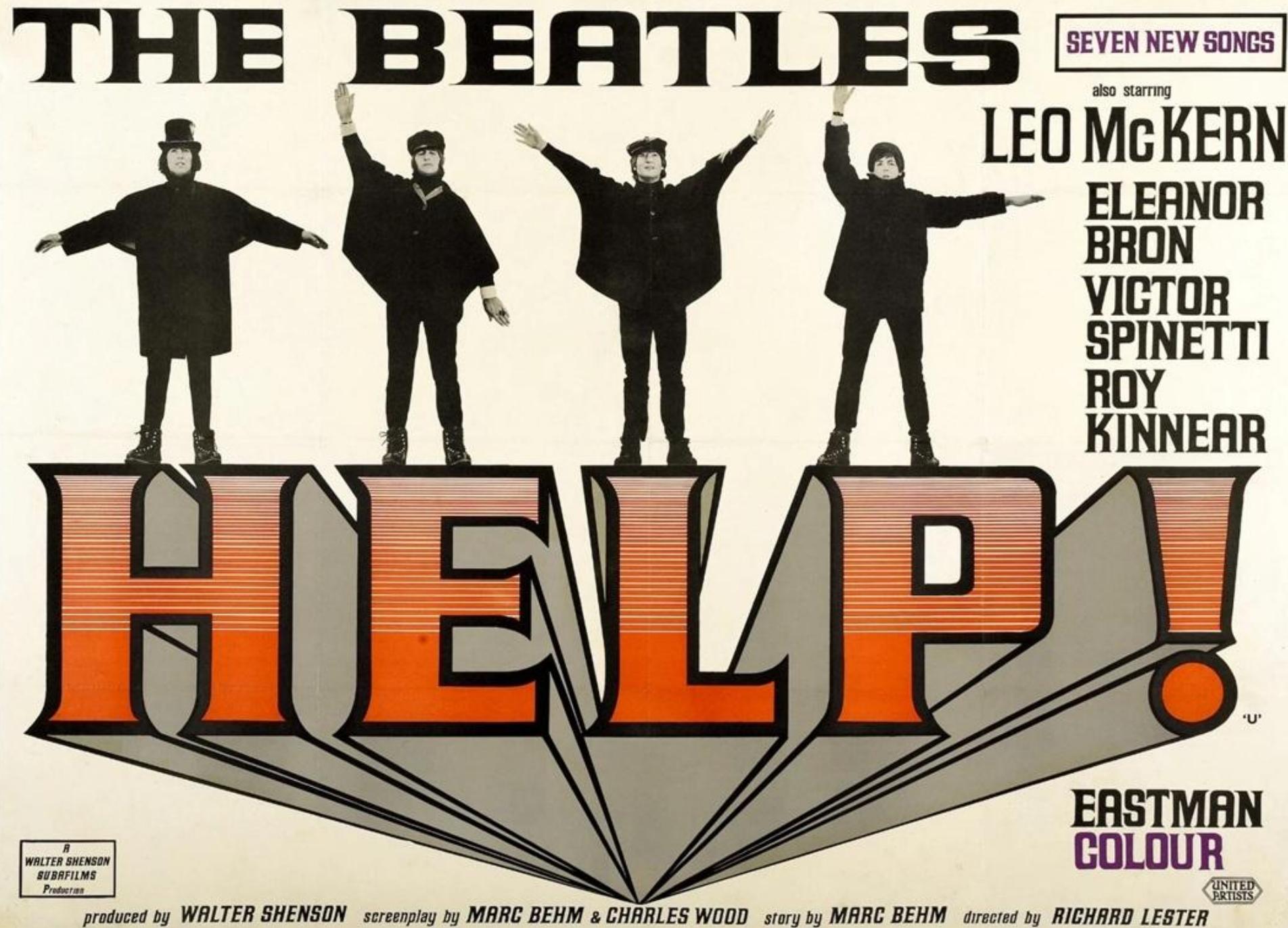
Line 2, Column 24

UTF-8

RESULTS

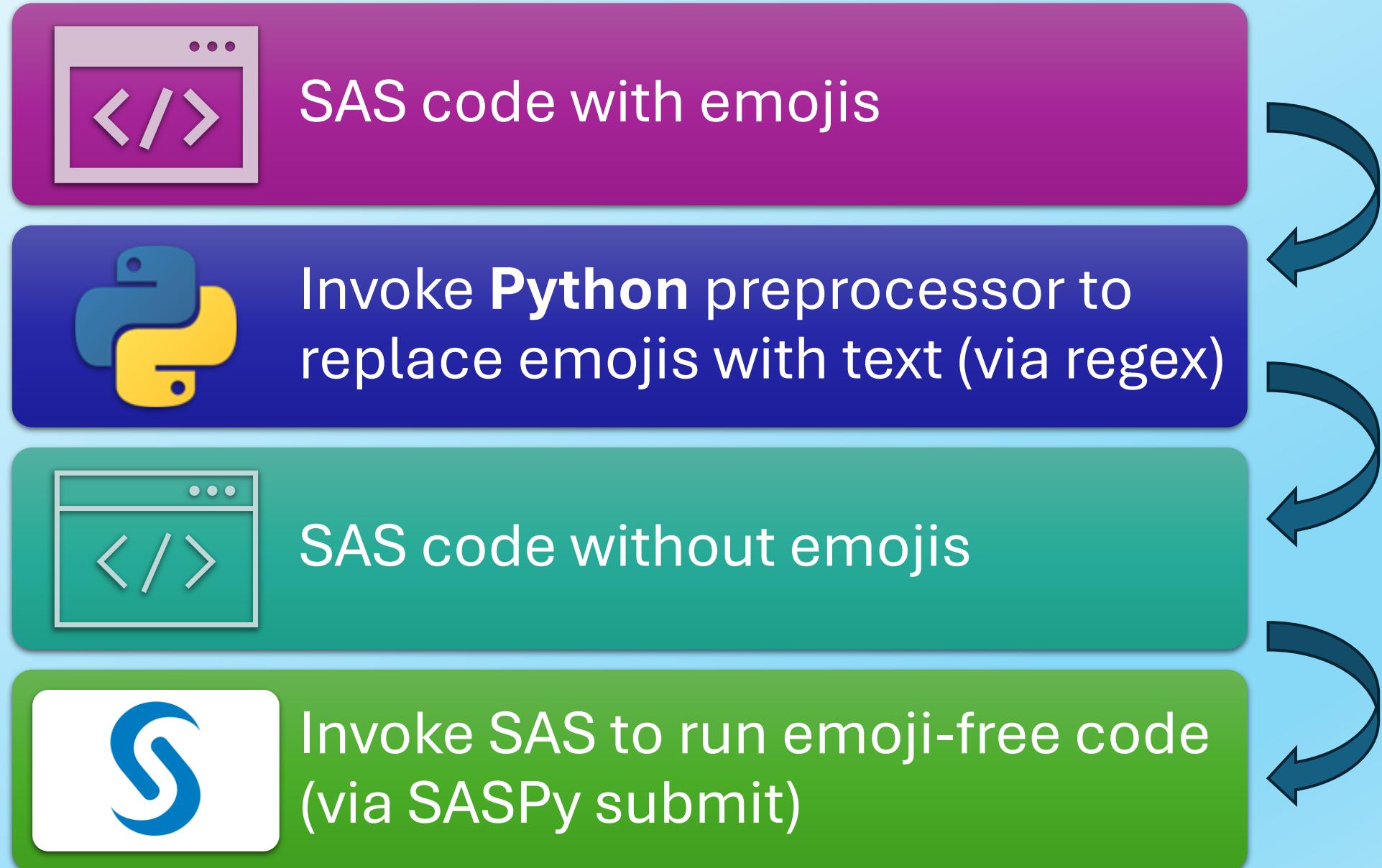
▶ Table of Contents

A Python Preprocessor?



produced by **WALTER SHENSON** screenplay by **MARC BEHM & CHARLES WOOD** story by **MARC BEHM** directed by **RICHARD LESTER**

Plan C – A Python Preprocessor



A Quick SAS-to-Python Port

- Convert PROC FORMAT to a Python dictionary
- Use **regex** package with **/X** option to get a list of characters and graphemes from SAS code
- Use map functionality + dictionary to replace emojis with text
- Use SASPy to submit emoji-free SAS code to SAS OnDemand for Academics for execution



JUNE 16, 1989

```
dict = { '🤔' : 'if', '✅' : 'then', '✖️' : 'else',  
        '❓' : 'if', '👍' : 'then', '👎' : 'else',  
        '💾' : 'data', '✂️' : 'substr',  
        '📅' : 'date', '🕒' : 'time',  
        '🏃' : 'run;', '🖨️' : 'print',  
        '🗃️' : 'create table', '🗑️' : 'drop table',  
        '🛒' : 'select', '⬅️' : 'from',  
        '🔗' : 'join', '🔀' : 'order by',  
        '⬇️' : 'desc', 'Σ' : 'group by'}
```

PYTHON REGEX EMOJI PARSING

```

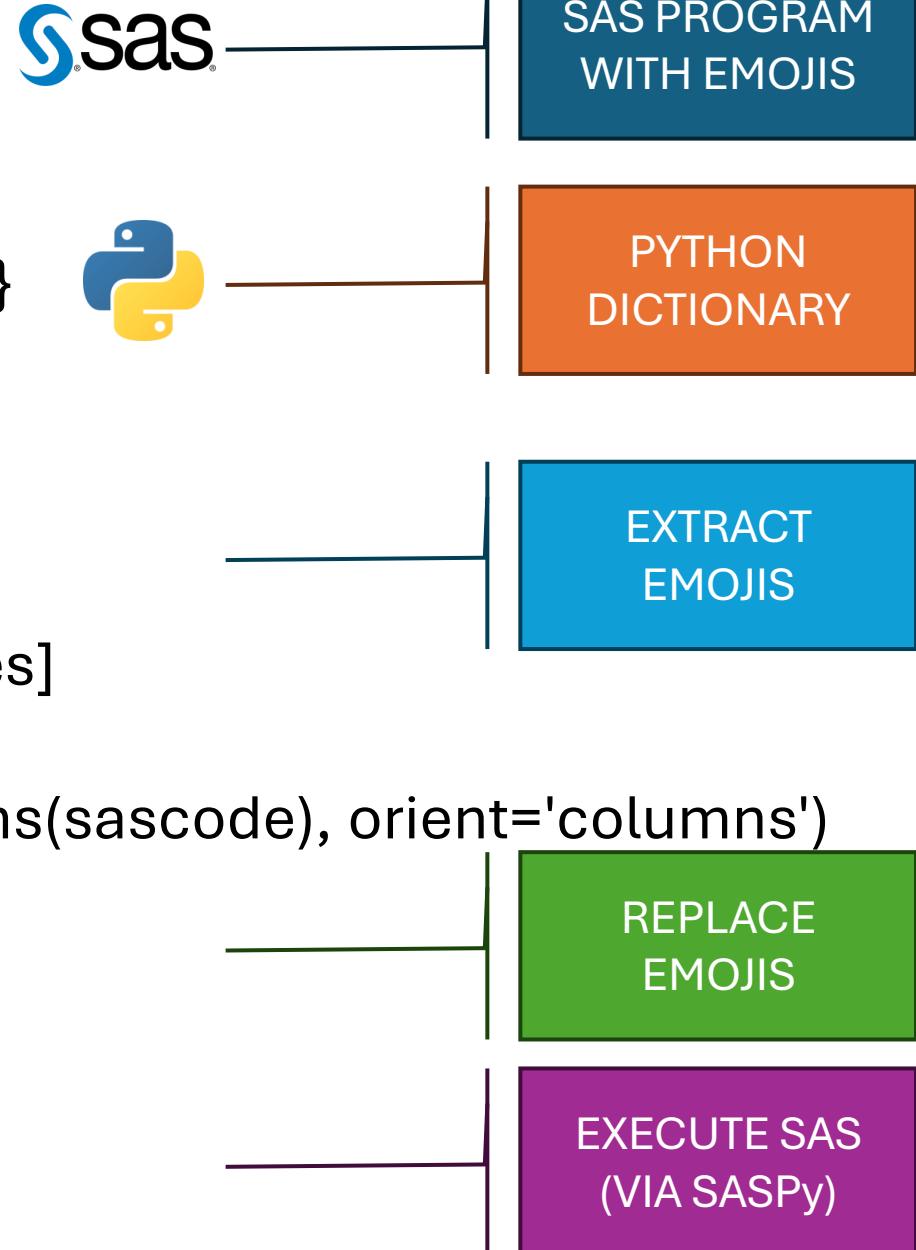
import regex, pandas as pd
dict = {'🤔': 'if', '✓': 'then', '✖': 'else', '🏃': 'run;'}
str="🤔 a=b ✓ c=d; ✖ e=f; ⚡"
df = pd.DataFrame([match.group() for match in regex.finditer(r"\X", str)],
                  columns=['grapheme'])
df['text'] = df['grapheme'].map(dict)
df['text'].fillna(df['grapheme'], inplace=True)
df

```

#	Grapheme	Text	#	Grapheme	Text	#	Grapheme	Text	#	Grapheme	Text
0	🤔	if	6	✓	then	12			18	;	;
1			7			13	✖	else	19		
2	a	a	8	c	c	14			20	🏃	run;
3	=	=	9	=	=	15	e	e			
4	b	b	10	d	d	16	=	=			
5			11	;	;	17	f	f			

PYTHON PREPROCESSOR (CODE)

```
sascode=""  
data _null_;  
    a=b ✅ c=d; ❌ e=f; 🏃  
"  
dict = {'👉': 'if', '✅': 'then', '❌': 'else', '🏃': 'run;'}  
  
import regex, json, pandas as pd  
def grapheme_positions(text):  
    matches = regex.finditer(r'\X', text)  
    return [{‘grapheme’: m.group()} for m in matches]  
  
df = pd.DataFrame.from_dict(grapheme_positions(sascode), orient='columns')  
df[‘replstr’] = df[‘grapheme’].map(dict)  
df[‘replstr’].fillna(df[‘grapheme’], inplace=True)  
sascodeout="".join(df[“replstr”].tolist());  
  
sas_session.submitLST(sascodeout)
```



WussPythonDemo.ipynb ×

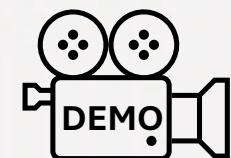
⚙️ ⚡ ⋮

C: > Users > tedco > OneDrive > Documents > SAS2023 > WussPythonDemo.ipynb > sascode=""

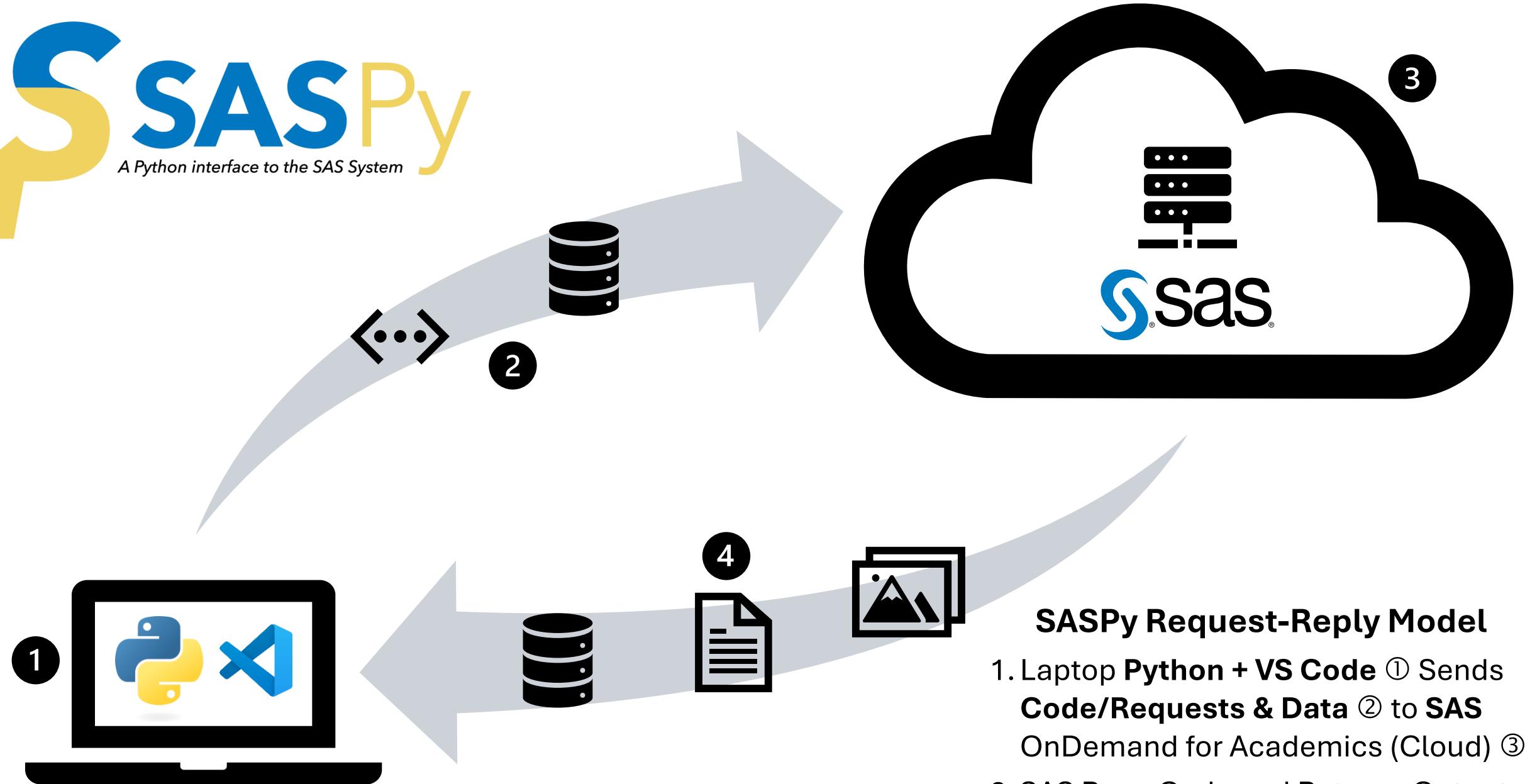
❖ Generate + Code + Markdown | ▶ Run All ⏪ Restart ⌂ Clear All Outputs ⋮

base (Python 3.10.9)

```
sascode='''  
data _null_;  
  😰 a=b ✓ c=d; ✗ e=f; 🏃  
...  
dict = {'😰': 'if', '✓': 'then', '✗': 'else', '🏃': 'run;'}  
  
import regex, json, pandas as pd      # Extract emojis from text;  
def grapheme_positions(text):  
    matches = regex.finditer(r'\X', text)  
    return [{grapheme: m.group()} for m in matches]  
                                # Replace emojis with text  
df = pd.DataFrame.from_dict(grapheme_positions(sascode), orient='columns')  
df['replstr'] = df['grapheme'].map(dict)  
df['replstr'].fillna(df['grapheme'], inplace=True)  
sascodeout="".join(df["replstr"].tolist());  
sas_session.submitLST(sascodeout)    # Run SAS Code using SASPy  
                                    # Show input & output SAS code  
print(sascode,sascodeout)
```



[12]



SASPy Request-Reply Model

1. Laptop **Python + VS Code** ① Sends **Code/Requests & Data** ② to **SAS OnDemand for Academics (Cloud)** ③
2. SAS Runs Code and Returns Output (**Images, Reports, Data**) ④ to Python

Contact Information



Ted Conway
Chicago, IL
ted.j.conway@gmail.com
[@vivasasvegas](https://twitter.com/vivasasvegas) (Twitter/X)



```
proc format;          * Define emojis and associated text;
value $emojif '💾'='data' '🤔'='if' '✓'='then' '✖'='else' '🏃'='run';
data _null_;         * Read in code, replace emojis w/associated text;
length txt varchar(1024);
infile "/home/ted.conway/test_pgm_in.sas" irecl=32767 recfm=n truncover length=l;
file "/home/ted.conway/test_pgm_out.sas";
input pgm $varying32767. l;
do c=1 to klength(pgm);
  txt=put(ksubstr(pgm,c,1),$emojif.);
  txt_len=length(txt);
  put txt $varying. txt_len@;
end;
run;                 * Run preprocessed, emoji-free code;
%include "/home/ted.conway/test_pgm_out.sas";
                     * Create SAS dataset to show input & output SAS code;
filename pgms '~/test_pgm_*.sas';
data code_in_out; infile pgms filename=fn; input; code=_infile_;
```



```
import saspy                                # Import SASPy package
sas_session = saspy.SASsession()              # Start SAS OnDemand for Academics session

sascode=""                                     * Test SAS code;
data _null_;
  a=b  ✓  c=d; ✗  e=f; 🏃
"""

dict = {'❗': 'if', '✓': 'then', '✗': 'else', '🏃': 'run;'}

import regex, json, pandas as pd             # Extract emojis from text;
def grapheme_positions(text):
    matches = regex.finditer(r'\X', text)
    return [{grapheme: m.group()} for m in matches] # Replace emojis with text

df = pd.DataFrame.from_dict(grapheme_positions(sascode), orient='columns')
df['replstr'] = df['grapheme'].map(dict)
df['replstr'].fillna(df['grapheme'], inplace=True)
sascodeout="" .join(df["replstr"].tolist());
sas_session.submitLST(sascodeout)            # Run SAS Code using SASPy
print(sascode,sascodeout)                   # Show input & output SAS code
```

